Andrei Pogor

Licenciado em Ciências da Engenharia Eletrotécnica
e de Computadores

Técnicas de Aprendizagem Automática para a Identificação de Jogadas de Golfe

Dissertação para Obtenção do Grau de Mestre em
Engenharia Eletrotécnica e de Computadores

Orientador: José Manuel Fonseca,
Professor Auxiliar com Agregação
Universidade Nova de Lisboa

Júri:

Presidente: André Damas Mora, FCT-UNL

Arguente: Luís Brito Palma, FCT-UNL

Vogal: José Manuel Fonseca, FCT-UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2017

Técnicas de Aprendizagem Automática para a Identificação de Jogadas de Golfe

Dedico este trabalho aos meus pais e a todas as pessoas que acreditaram em mim

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao Professor José Manuel Fonseca pelos conhecimentos transmitidos, pela sua disponibilidade, pela orientação e conselhos ao longo de toda a dissertação e ainda pela paciência e amizade.

Também quero agradecer ao Nuno Duro pela oportunidade de trabalhar na empresa Bluecover e pelas novas experiências, pelo apoio e pelas insistências para o desenvolvimento desta dissertação.

Quero agradeço aos meus colegas e amigos de curso Tiago Santos, Vasco Brito, Miguel Santos, Gonçalo Leandro pelo apoio e conselhos na escrita deste documento. Gostaria ainda de agradecer a todos os outros colegas que me acompanharam ao longo deste percurso.

Agradeço em especial à Cristina Petrari pelo apoio e pelo carinho que me transmite todos os dias.

Um reconhecimento profundo à minha família pela coragem e incentivo. Aos meus pais pelo apoio incondicional ao longo destes anos, pelos sacrifícios feitos ao longo da minha educação e formação profissional, sem os quais não seria possível concluir esta etapa da minha vida.

Por último, mas não menos importante, queria agradecer à Faculdade de Ciências e Tecnologia da UNL por me ter acolhido nestes últimos seis anos e por me darem oportunidade de fazer Erasmus. Um agradecimento também ao país pelo seu acolhimento.

Resumo

Com esta dissertação propõe-se desenvolver um acessório inteligente para ser utilizado por golfistas com o objetivo de tornar o jogo de golfe mais atrativo. O acessório inteligente a desenvolver consiste numa pulseira eletrónica que se interliga com um telemóvel inteligente e que informa o utilizador da duração da cada movimento efetuado durante o movimento completo de uma jogada. Este acessório permite também avaliar a rotação e a aceleração máxima no momento do batimento do taco na bola, analisar os tipos de movimentos e por fim, comparar com os movimentos anteriores ou até mesmo com movimentos de outros jogadores.

A pulseira desenvolvida neste projeto inclui sensores que medem a rotação angular e a aceleração nos três eixos, o Bluetooth que são ligados a um microprocessador que contém a lógica de controlo embutido e que tem como funções principais controlar os diversos dispositivos da pulseira, receber comandos do telemóvel inteligente e enviar-lhe os dados dos sensores para posterior análise.

O telemóvel inteligente tem instalado um programa, desenvolvido anteriormente pela empresa denominada Bluecover, que permite ao utilizador enviar comandos para a pulseira, e por fim, receber os dados da pulseira e guardá-los num ficheiro de formato Excel.

Por fim, são desenvolvidos dois algoritmos: Um que identifica o movimento de golfe e extrai as suas principais características e um outro que identifica o tipo de movimento usando um algoritmo de classificação baseado em árvores de decisão e em redes neuronais.

Palavras-chaves: Identificação, Movimento de Golfe, Características, Árvores de Decisão, Redes Neuronais, Detecção Automática.

Abstract

This dissertation aims to develop a smart-device to be used by golfers during a game in order to make it more compelling. This device has the ability to measure the duration of each movement within a golf-shot and their acceleration. After the data acquisition process, this device compares the measured data with previously acquired data from both the user and a database of existing users. This device is a wristband connected via Bluetooth to a smartphone.

The wristband comprises a set of different sensors, the Bluetooth that are connected to a microprocessor which has embedded the controller logic. This wristband will then send the data to the smartphone.

The smartphone has an app installed previously developed by Bluecover company, which is responsible to coordinate instructions sent to and from the wristband. This app will also store the data sent from the wristband in an Excel file.

Last but not least two algorithms were developed. One extracts the golf-shot main features and the other classifies the shot using a classification algorithm with decision trees and neural networks.

Keywords: Identification, Golf Swing, Features, Decision Trees, Neural Networks, Automatic Detection.

Simbologia

RMS: Valor Quadrático Médio, do inglês *Root-Mean-Square*

I2C: Circuito Inter-integrado, do inglês *Inter-Integrated Circuit*

PWM: Modulação por Largura de Pulsos, do inglês *Pulse-Width Modulation*

IDE: Ambiente de Desenvolvimento Integrado, do inglês *Integrated Development Environment*

DAC: Conversor digital para analógico

GPS: Sistema de Posicionamento Global, do inglês *Global Positioning System*

USB: Porta Universal, do inglês *Universal Serial Bus*

Uart: Receptor/Transmissor Assíncrono Univeral, do inglês *Universal Asynchronos Receiver/Transmitter*

Swing: Movimento de golfe

DC: Corrente Contínua, do inglês *Direct Current*

Dps: Graus por segundo

g: Unidade gravitacional

Sampling Rate: Taxa de amostragem

HCP: *Handicap*

σ: Desvio padrão

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Motivação

Em primeiro lugar, o desenvolvimento desta dissertação tem por objetivo introduzir o fator tecnológico em atividades do dia-a-dia e desta forma proporcionar um maior prazer e uma aprendizagem mais rápida. Assim surgiu a oportunidade de a explorar na modalidade do golfe.

Este desporto pratica-se num campo relvado, ao ar livre. Uma bola de pequena dimensão é projetada por um taco e utilizando o menor número de tacadas possíveis, esta deve ser colocada num determinado orifício no relvado. A prática deste desporto tem grandes benefícios para a saúde física e mental sendo apropriada para pessoas de todas as idades e géneros (Evans & Tuttle, 2015).

Têm sido desenvolvidos muitos projetos tecnológicos na área do golfe para o apoio à prática deste desporto com o objetivo de o tornar este desporto mais atrativo. Como exemplo destes avanços tecnológicos destacam-se o mecanismo de informar sobre campos existentes, informar as pessoas que estão a frequentá-lo, ou o uso de dispositivos extremamente caros (18.000$) (TrackMan, 2016) para monitorizar o movimento do golfe desde o início até o fim. Sumarizando, todas estas tecnologias visam o enriquecimento informativo e educativo desta modalidade atraindo deste modo um maior número de pessoas.

A grande maioria dos jogos de golfe ocorre sem supervisão de um árbitro ou observador. Essa falta de monitorização dos jogos torna muitas vezes difícil ao jogador visualizar a sua evolução e consequentemente o jogador vai cometendo erros sem que seja corrigido.

Assim sendo, surge a oportunidade de pensar numa tecnologia que ajude o jogador na sua evolução constituindo uma forma de apreender autonomamente.

Resumindo, esta dissertação tem por objetivo implementar uma pulseira a partir de materiais de custo reduzido que poderá conectar-se com um telemóvel inteligente, e este irá armazenar todas as informações relevantes para o jogador e para o seu progresso como, por exemplo, mostrar os tempos que demorou a realizar um movimento completo de golfe, mostrar graficamente o percurso do movimento e identificar o tipo de movimento.

1.2 Âmbito, objetivos e contribuições

O principal objetivo desta dissertação consiste no desenvolvimento de *Hardware* e *Software* utilizando tecnologia recente de baixo custo para ajudar a melhorar e aperfeiçoar os movimentos de golfe durante o jogo.

Esta dissertação contribui para o golfe em duas vertentes: informativa e educativa. Quanto à primeira vertente, atrai mais pessoas para que estas tenham uma maior iniciativa de experimentar este desporto. A segunda vertente diz respeito à necessidade de o jogador praticar golfe de uma forma autónoma, porém proporcionando-lhe o conhecimento do percurso e do tipo de movimento efetuado, tais como, *Full swing* ou *Pitching*, sendo o *Swing* o nome que se dá ao movimento do golfista.

Resumindo, esta tecnologia permite visualizar a evolução e ainda comparar com os movimentos de outros jogadores. Desta forma, conduz a resultados positivos num menor espaço de tempo pois incentiva à competição.

Quanto aos dados extraídos, estes contribuem para a análise de vários pontos importantes que se podem extrair de um movimento completo de golfe, tais como, o tempo de preparação, a duração do movimento, a aceleração e a rotação máxima no momento do batimento com a bola, entre outros.

Os dados recolhidos por esta tecnologia vão dar conhecimento ao golfista da qualidade do seu movimento, permitindo-lhe desta forma avaliar quais os passos para melhorar o seu desempenho. Para além disso, permite-lhe ver a sua evolução na globalidade.

Para finalizar, esta dissertação pressupõe o desenvolvimento de uma tecnologia composta por *software* (algoritmo de detecção de movimentos de golfe, extração de características e análise dos algoritmos de classificação) e hardware de baixo custo capaz de enfrentar os desafios do golfe e assim ajudar na educação e divulgação desta modalidade.

1.3 Estrutura da dissertação

Esta dissertação está dividida em 5 capítulos, incluindo este, com a seguinte descrição:

No capítulo 1 são apresentadas as motivações, âmbito, objetivos e contribuições da dissertação.

O capítulo 2 aborda os conceitos básicos do golfe e os trabalhos desenvolvidos anteriormente no âmbito desta dissertação.

O capítulo 3 tem como objetivo dar a conhecer as ferramentas necessárias para a implementação da pulseira, descrevendo todos os passos para a criação do algoritmo de detecção de movimentos de golfe, extração de características e analise dos algoritmos de classificação.

No capítulo 4 foram realizadas simulações dos algoritmos de classificação e realizado um estudo de comparação dos resultados experimentais.

O capítulo 5 contém as conclusões sobre toda a dissertação desenvolvida e o trabalho a desenvolver no futuro.

Por fim, surgem referências e anexos, contendo a programação da pulseira, a programação de detecção de movimentos de golfe e extração de características, e a programação do classificador, toda em linguagem C.

2 Estado da Arte

Este capítulo tem como objetivo introduzir o suporte teórico básico para a definição dos conceitos aplicados no golfe e assim compreender a terminologia e as noções essenciais para o correto desenvolvimento dos algoritmos no decorrer desta dissertação.

Posteriormente serão mencionados os trabalhos relacionados na área de golfe, pois estes mostram outra forma de abordar o trabalho que será desenvolvido nesta dissertação.

2.1 Introdução ao golfe

O golfe tem origem na Escócia do século XV onde inicialmente se usava um bastão de madeira para bater em objetos pequenos de forma redonda.

Inicialmente era usada uma bola de madeira que evoluiu significativamente tendo passado por muitas modificações tecnológicas ao longo da história. Uma das principais foi a introdução de pequenas concavidades na sua superfície que melhoram em muito a sua precisão.

A partir do século XVI, a modalidade começou a expandir-se por toda a Europa, principalmente junto das classe nobres. Inicialmente, as regras no continente americano e europeu eram diferentes, porém estas foram uniformizadas e generalizadas em 1951 (Infopedia, 2016).

Desde o início que o golfe é praticado em vastas áreas de terreno, contendo obstáculos naturais, tais como: lagos, dunas, areais e árvores (Infopedia, 2016). O objetivo principal do golfe consiste em colocar a bola nos 18 orifícios do campo com o menor número possível de tacadas (Tacadas, 2013).

Para finalizar esta breve introdução histórica, é possível verificar que ao longo dos tempos esta modalidade tem vindo a crescer a todos os níveis e que ainda é possível aplicar novas tecnologias que visem uma melhor compreensão e aprendizagem.

Conceitos básicos

Inicialmente torna-se necessário entender o conjunto de equipamentos utilizados para praticar esta modalidade desportiva e perceber a sua importância. Seguidamente é necessário entender os termos utilizados, como por exemplo, o *handicap*, a descrição dos tempos e os principais tipos de movimentos encontrados no golfe (Rules, 2012).

O equipamento mais importante no golfe é o taco. Quanto à sua constituição este é formado por uma pega, uma vareta e uma cabeça, (ver Figura 2.1).

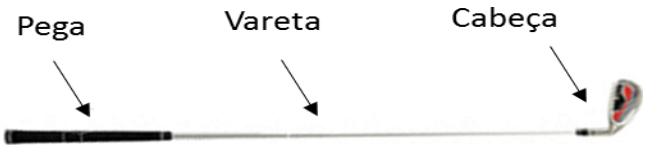


Figura 2.1: Constituição do taco de golfe.

Para estes existem vários termos, tendo em conta o material de fabrico: *Drivers*/Madeiras, Híbridos, Ferros e *Putter*, Figura 2.2.

Figura 2.2: Tacos de golfe (Tacadas, 2013).

Cada taco, na sua arquitetura, é composto por um conjunto de variantes tais como o ângulo entre a vareta do taco e a horizontal, o ângulo entre a face do taco e a vertical e o comprimento da vareta. Há vários tipos de tacos com especificações distintas o que obriga o jogador a usar vários tacos durante um jogo. Porém, cada jogador só pode levar para o campo um máximo de 14 tacos (Nuno Duro, 2015).

Como já foi citado existem vários tipos de tacos. Assim sendo, iremos seguidamente analisá-los.

Inicialmente os tacos de madeira têm como finalidade alcançar distâncias mais longas e são os preferidos dos golfistas.

Os de ferro são usados para alcançar distâncias mais curtas e comparando com os anteriores, oferecem uma maior precisão para quando se quer efetuar as últimas tacadas.

Por fim, os *putters* são usados normalmente para a conclusão de um jogo, por terem uma precisão superior aos de ferro (Tacadas, 2013).

Para além do taco, as bolas são o outro equipamento essencial para a prática desta modalidade. Estas têm um desenho muito preciso, um peso de exatamente 45,93 gramas e um diâmetro 42,67mm (Golfe Jogar, 2016). São, por norma, brancas, mas podem ser encontradas noutras cores e apresentam ligeiras concavidades para melhorar a aerodinâmica reduzindo o atrito entre a bola e o ar, Figura 2.3.

Figura 2.3: Bola de golfe (Golfe Jogar, 2016).

Apresentaremos, de seguida alguns termos essenciais a este desporto.

Quanto ao *handicap*, este é usado para determinar a habilidade de um jogador de golfe, e tem de ser periodicamente ajustado, a fim de manter o equilíbrio e a competitividade de jogo entre os praticantes. Para as senhoras vai de 0 a 36 e para os homens, vai de 0 a 28. Os jogadores profissionais não têm *handicap*, ou seja, jogam sempre com *handicap* de 0. Resumindo, quanto mais baixo for o *handicap* do jogador maior será a sua habilidade (Nuno Duro, 2015).

Os tempos que se podem extrair durante um movimento completo de golfe estão descritos na Tabela 2.1.

Posteriormente, na Tabela 2.2 são descritos os principais tipos de movimentos de golfe, as suas características, os tacos utilizados e ainda a distância alcançável. Por último, um dos termos mais utilizados no golfe é o *Swing*, sendo este o nome que se dá ao movimento de golfe.

Tabela 2.1: Os tempos do Swing.

Tempo	Descrição
Backswing Time	Duração do movimento para trás, ou preparação, (ver Figura 2.4).
Downswing Time	Duração do movimento do batimento da bola, (ver Figura 2.5).
Follow-Through Time	Duração da desaceleração, (ver Figura 2.6).

Figura 2.4: *Backswing*.

Figura 2.5: *Downswing*.

Figura 2.6: *Follow-Through*.

Tabela 2.2: Lista de movimentos.

Swings (*Clubs*/Tacos)	Características
Full swing (Short Iron)	Movimento completo de *Swing* em que são usados os ferros 8i, 9i, PW, SW. Este movimento permite alcançar curtas distancias (de 60 a 110 metros), ou para obstáculos altos como uma árvore na proximidade.
Full swing (Long Iron)	Movimento completo de Swing em que são usados os ferros 7i a 3i e híbrido. Para longas distâncias, pode ir de 100 a 160 metros, com ângulo baixo de lançamento.
Full swing (Driver)	Movimento completo de Swing em que são usados os tacos *drive* e *woods*. Usado para obter maiores distâncias que podem ir além dos 200 metros.
*Pitching (*60 metros*)*	Movimento mais curto do que *Full Swing* (3/4 de *Full Swing*) para aproximação de buraco, tipicamente para passar por cima dos obstáculos. O objetivo é alcançar uma distância de 60 metros.
Pitching (30 metros)	Movimento mais curto do que *Full Swing* (3/4 de *Full Swing*) para aproximação de buraco, tipicamente para passar por cima dos obstáculos. O objetivo é alcançar uma distância de 30 metros.
Bunker (5 metros)	Movimento usado para tirar a bola de um depósito de areia, e é para alcançar curtas distâncias, pode ir de 5 a 20 metros.
Chipping (2,5 metros)	Movimento jogado perto do green, geralmente em curtas distâncias de colocação, o que resulta na bola saltando no ar, em seguida, bater no chão e rolando para a frente, o objetivo é alcançar uma distância de 2,5 metros.
Chipping (7,5 metros)	Movimento jogado perto do buraco, geralmente em curtas distâncias de colocação, o que resulta na bola estalando no ar, em seguida, bater no chão e rolando para a frente, o objetivo é alcançar uma distância de 7,5 metros.
Short Putts (Putter 0,5 metros)	Movimento suave que normalmente é para concluir o jogo, é usado o putter. Para alcançar curtas distâncias com uma boa precisão, o objetivo é alcançar 0,5 metros.

Medium Putts (Putter 4 metros)	Movimento suave que normalmente é para concluir o jogo, é usado o *putter*. Para alcançar curtas distâncias com uma boa precisão o objetivo é alcançar 4 metros.
Long Putts (Putter 15 metros)	Movimento suave que normalmente é para concluir o jogo, é usado o *putter*. Para alcançar curtas distâncias com uma boa precisão, o objetivo é alcançar 15 metros.

2.2 Trabalhos relacionados

Para poder avançar com um novo projeto, é importante pesquisar os trabalhos já desenvolvidos dentro da mesma área, percebendo se o novo projeto vai ter alguma contribuição ou não para a área. O Golfshot (Golfeshot, 2016) e o Trackman (TrackMan, 2016) são dois projetos tecnológicos desenvolvidos para golfe. O projeto "Classificação de atividade humana" (Bunkheila, 2016), embora não tenha sido desenvolvido com um desporto como objetivo revelou-se muito útil como inspiração para o desenvolvimento desta dissertação.

2.2.1 Golfshot

Golfshot é uma aplicação para golfe compatível com o sistema Android e iOS que tem a capacidade de exibir a distância a que a pessoa se encontra do orifício e quantas tacadas precisa de dar para alcançá-lo conforme o seu nível de habilidade Figura 2.7.

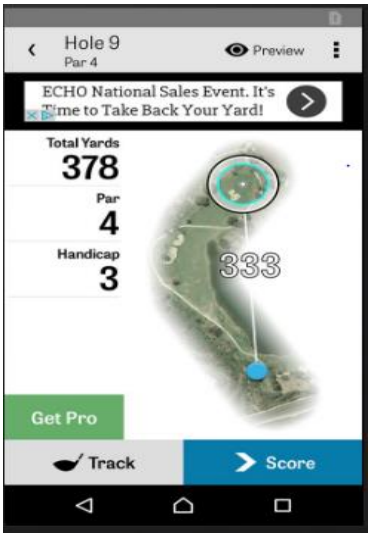


Figura 2.7: Distância do centro do buraco (Golfeshot, 2016).

Posteriormente, esta aplicação tem a capacidade de gravar as posições de cada tacada efetuada e também monitorizar a distância que o jogador percorreu. Existem, porém, diversas versões. A versão profissional é paga e tem mais funcionalidades, tais como, mostrar em tempo real os perigos existentes, recomendação dos tacos baseados no desempenho do golfista, e por fim, uma pré-visualização em 3D.

A Golfshot tem uma versão que pode ser instalada em relógios inteligentes, tendo neste caso apenas duas funcionalidades: mostra a distância ao orifício e algumas informações básicas, Figura 2.8 (Golfeshot, 2016).

Figura 2.8: Relógio Inteligente (Golfeshot, 2016).

Por fim, esta aplicação baseia-se num *software* que usa recursos do GPS, do telemóvel inteligente, para fazer todas as monitorizações necessárias. Porém, nenhuma das funcionalidades analisa o movimento realizado pelo golfista não possibilitando a aprendizagem efetiva do jogador.

2.2.2 Trackman

Trackman, Figura 2.9, é um dispositivo eletrónico que analisa o que acontece quando a bola de golfe é atingida por um taco antes, durante e após o movimento do mesmo.

A última versão contém dois radares, um para medir o voo da bola até à zona de aterragem, onde serão medidos vários parâmetros, tais como: ângulo de lançamento, taxa de rotação, velocidade do taco, curvatura e outras medidas.

Enquanto o outro radar, utiliza também uma tecnologia baseada numa câmara para gravar o momento do impacto e uma visualização 3D do voo da bola. Os dados recolhidos pela Trackman são enviados via *Wi-Fi* para um computador ou para um telemóvel inteligente (TrackMan, 2016).

Figura 2.9: Trackman (TrackMan, 2016).

Este equipamento consegue medir tudo o que é necessário para saber o que se deve melhorar para diminuir a pontuação, identificando os pontos fortes e fracos, e observar a diferença entre jogadores (Figura 2.10). O equipamento pode ser adquirido a partir de 18.000$ dólares americanos.

Figura 2.10: Comparação entre dois jogadores de golfe (TrackMan, 2016).

Com este dispositivo é possível analisar um movimento de golfe com muitos detalhes, mas não é possível identificar o tipo de movimento. Embora seja muito preciso em termos de sensores e aquisição de dados, o preço é muito elevado, o que torna difícil a sua aquisição.

2.2.3 Classificação de atividade humana

Este tipo de classificação monitoriza a atividade humana baseada na leitura dos estímulos realizada pelos sensores do telemóvel inteligente.

O objetivo do trabalho consistiu na construção de um algoritmo que fosse capaz de identificar automaticamente as atividades medidas pelo acelerómetro. Desta forma, foram capturados seis tipos de atividades físicas, tais como andar, subir e descer escadas, estar sentado, estar em pé e permanecer deitado (Bunkheila, 2016).

A ideia consistia em ter muitas amostras para cada atividade e de seguida retirar as características de cada sinal para assim desenvolver um algoritmo da classificação utilizando as redes neuronais (Bunkheila, 2016). Em primeiro lugar, são analisados os sinais gerados por cada atividade ao longo do tempo, Figura 2.11.

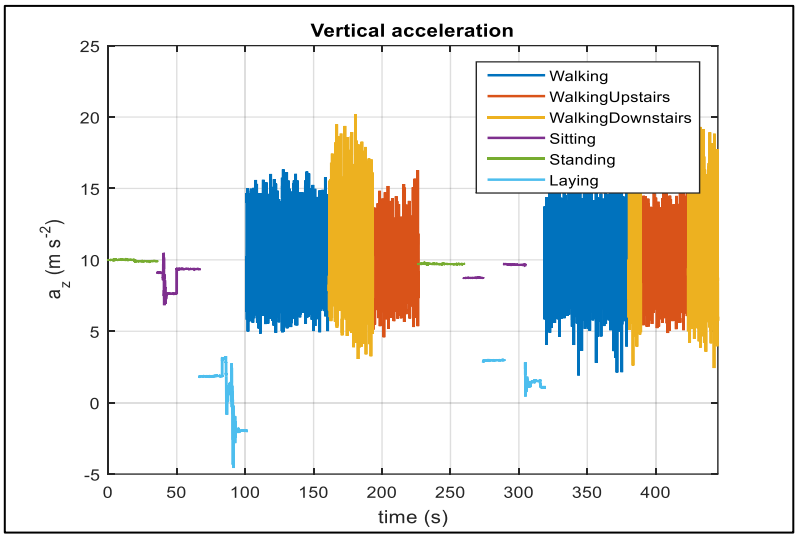


Figura 2.11: Sinal do eixo z do acelerómetro (Bunkheila, 2016).

De seguida, são procuradas as diferenças para cada sinal gerado pelas atividades efetuadas pelo utilizador. Por exemplo, o sinal gerado pelo andar é diferente do sinal gerado quando se está sentado, sendo porém muito similar ao subir as escadas, como se pode ver na Figura 2.12. Desta forma podemos concluir que atividades diferentes podem gerar acelerações idênticas, surgindo assim a necessidade de encontrar outra váriavel para as distinguir.

Posteriormente, foi necessário encontrar outros aspetos que possam diferenciar os sinais gerados pelas atividades.

Inicialmente é efetuada uma filtragem passa-banda para se retirar as baixas e as altas frequências e de seguida uma passagem do domínio do tempo para o domínio da frequência (Figura 2.13). Com esta operação, pode-se facilmente diferenciar entre a frequência do andar e a de subir escadas.

Para caracterizar as frequências dos sinais gerados pelas atividades foi necessário encontrar os máximos dos 8 picos, afastados uns dos outros pelo menos 0,25Hz e com um determinado valor de proeminência, Figura 2.14.

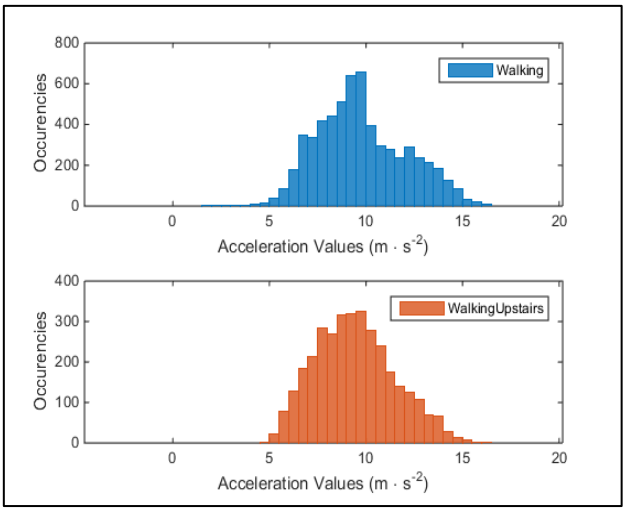


Figura 2.12: A diferença entre andar e subir escadas (Bunkheila, 2016).

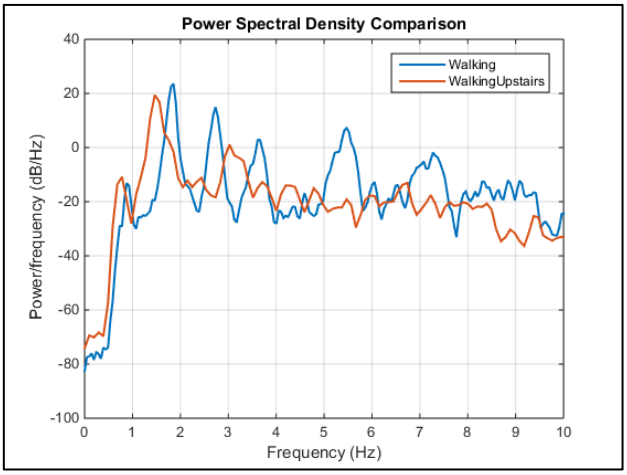


Figura 2.13: Aspeto de frequência do andar e subir escadas (Bunkheila, 2016).

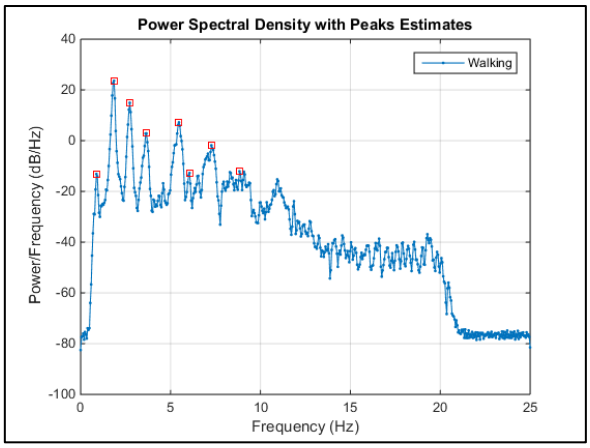


Figura 2.14: Os máximos 8 picos (Bunkheila, 2016).

Seguidamente, são extraídas todas as características do sinal gerado pelas atividades e pela frequência do mesmo, como por exemplo o valor médio, a raiz quadrada da média (RMS), a altura do pico principal, a altura e a posição dos outros 7 picos, a potência total em 5 picos adjacentes e as bandas de frequências pré-definidas de cada eixo (Bunkheila, 2016).

Na fase seguinte, é atribuída uma classe para cada conjunto de características sendo essa classe associada a cada tipo de atividade exercida.

No momento em que a base de dados está completa com todas as características é desenvolvido um algoritmo da classificação utilizando redes neuronais.

A Figura 2.15 é a representação de uma matriz de confusão utilizada nos testes das redes neuronais, criada pelo *software* MatLab, onde foi desenvolvido todo o algoritmo de extração de características e a próprias redes neuronais.

Pela matriz da confusão é possível concluir vários aspetos, por exemplo para a *target* classe 1, dos 1722 cenários possíveis, 1615 amostras foram definidas como de classe 1, 73 como classe 2 e 34 de classe 3, respetivamente. Isto significa que a classe 1 pode ser confundida com a 2 e 3, porém este fenómeno não acontece com a classe 4, 5 e 6, para a *target* classe 2, dos 1544 cenários possíveis, 1411 amostras foram definidas como classe 2, 66 como classe 1, 62 como classe 3 e 5 de classe 5 respetivamente, isto significa que a classe 2 pode ser confundida com a 1, 3 e 5, e assim sucessivamente. Resumidamente, as redes neuronais tiveram 90,7% de sucesso e 9,3% de insucesso.

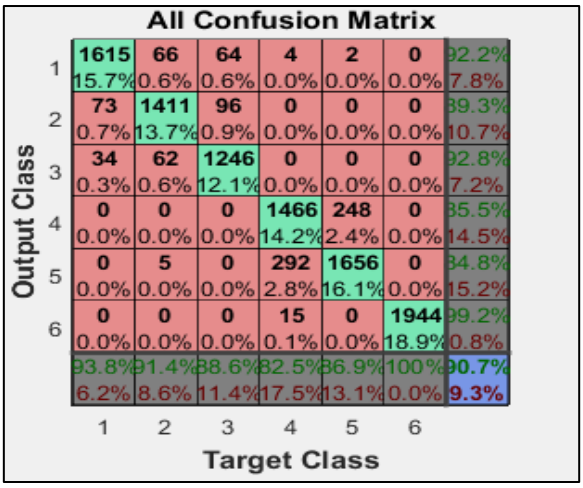


Figura 2.15: Matriz da confusão (Bunkheila, 2016).

Posteriormente, um método recorrendo a redes neuronais necessitam de recorrer a várias capturas de dados, isto é, este tipo de método necessita de ter treino, por forma a obter-se uma boa prática. Os vários treinos garantem que os resultados não serão influenciados pelos dados de treino anteriormente referidos (Bunkheila, 2016).

A Figura 2.16 mostra a matriz da confusão dos resultados dos testes para novas amostras, e mais uma vez, as redes neuronais tiveram uma boa resposta. Neste caso as redes neuronais obtiveram 91,7% de sucesso face a 8,3% de insucesso.

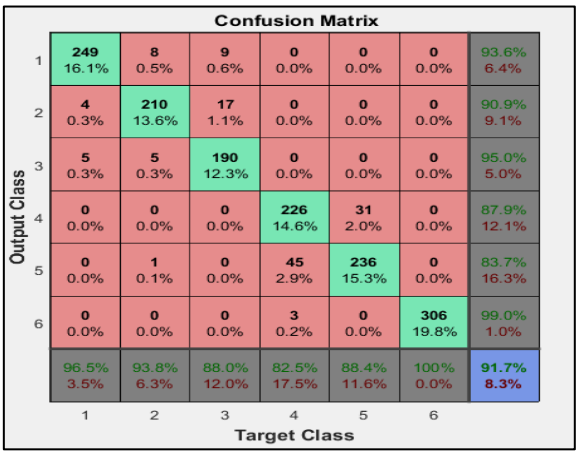


Figura 2.16: Matriz da confusão com amostras diferentes (Bunkheila, 2016).

Embora este trabalho não tenha sido projetado tendo em vista o golfe, a forma elegante e efetiva como efetua a extração de características e a classificação dos movimentos humanos constituiu uma inspiração para esta dissertação.

3 Metodologia

Após o estudo das técnicas, termos e do estado da arte relacionados com esta modalidade, é possível ajustar algoritmos por forma a obtermos um modelo de aprendizagem automática capaz de auxiliar o jogador de golfe.

Em primeiro lugar, torna-se necessário efetuar a leitura dos movimentos necessários, funcionando como entradas do sistema. Para a aquisição sensorial será necessário recorrer a um *hardware* composto por sensores que meçam a aceleração e a rotação angular do pulso (Stančin & Tomažič, 2013).

De seguida, para a implementação da lógica do sistema recorre-se a um microprocessador capaz de efetuar a leitura dos dados, processar os mesmos e de seguida enviar os dados para um telemóvel inteligente através de uma comunicação sem fios, para de seguida, serem manipulados.

Posteriormente, surge a necessidade de criar uma pulseira dotada de sensores e microprocessador. Apesar de já existir uma grande variedade de relógios inteligentes, com diversos sensores, permanece o problema de estes não serem capazes de fazer leituras de movimentos rápidos, como é o caso da tacada de um golfista. Existem ainda outros problemas como a falta de documentação e ainda o facto de cada fabricante ter as suas características o que torna difícil a criação de um algoritmo que seja válido para qualquer dispositivo.

Por fim, os dados são recebidos por um telemóvel inteligente e de seguida analisados dando assim uma resposta ao utilizador. Desta forma, é necessário desenvolver um algoritmo capaz de extrair as características necessárias de um movimento de golfe, em seguida, desenvolver um algoritmo de classificação do tipo de movimento.

3.1 Descrição do Hardware

Como primeiro passo para o desenvolvimento deste protótipo, temos de definir as limitações para a estrutura da pulseira equipada com o *hardware* customizado pois não podem causar um obstáculo físico para o desempenho do golfista.

Os seguintes requisitos são relativos à funcionalidade. Em primeiro lugar, a bateria deve ter capacidade para durar pelo menos um jogo completo, o microprocessador utilizado deve ser capaz de manipular os dados lidos pelos sensores e atuar de forma rápida, sendo também capaz de suportar protocolos de comunicação sem fios.

Para o microprocessador foi escolhido o Teensy 3.2 (PTRobotics, 2016d), devido ao custo reduzido, facilidade de programação e suporte para diversos protocolos de comunicação. Estas características são ótimas para o desenvolvimento de protótipos.

3.1.1 Arquitetura do Hardware da Pulseira

A arquitetura de *Hardware* da pulseira é composta por um microprocessador, sensores Bluetooth e uma bateria. A comunicação entre o microprocessador e o Bluetooth é feita por comunicação série. Quanto aos sensores, a comunicação é assegurada por I2C. O microprocessador é alimentado diretamente através da bateria, sendo os restantes elementos alimentados por um regulador de tensão que está incorporado no microprocessador, Figura 3.1. Cada componente da arquitetura de *Hardware* da pulseira é descrito nos subcapítulos seguintes.

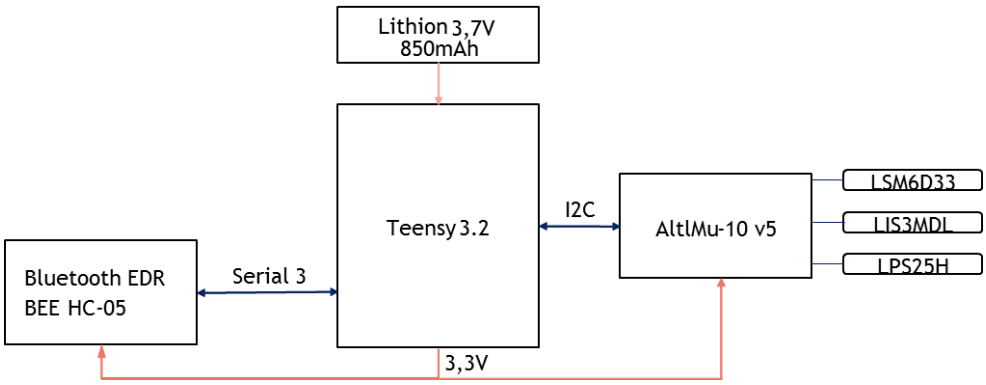


Figura 3.1: Arquitetura do *Hardware* da Pulseira.

3.1.2 Microprocessador

Como a pulseira tem que ser portátil e ter uma estrutura física de pequenas dimensões, foi escolhido como microprocessador o Teensy 3.2 (PTRobotics, 2016d) que apresenta pequenas dimensões e ainda um alto desempenho de processamento, o que torna esta plataforma ótima para este protótipo, Figura 3.2.

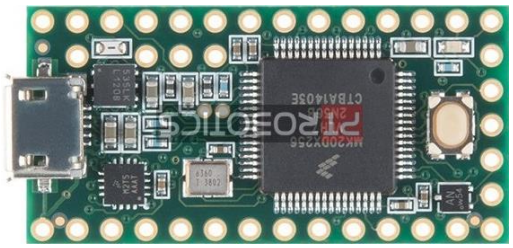


Figura 3.2: Teensy 3.2 (PTRobotics, 2016d).

O Teensy 3.2 é um microprocessador com 34 pinos digitais de entrada e saída, entre os quais, possui comunicação série e I2C, PWM, entre outros. Dispõe também um regulador de tensão de 3,3V, que é capaz de alimentar os outros módulos, tais como, os sensores e o Bluetooth.

Como ambiente de desenvolvimento de código é utilizado o IDE do Arduíno, onde é feita a compilação do código e a operação de *flash* para o target via USB. As principais características do microprocessador estão descritas na Tabela 3.1.

Tabela 3.1: Principais característica do Teensy 3.2 (PTRobotics, 2016d).

Dimensões: 18mm x 35mm x 4mm Processador: ARM 32 bit Núcleo: Cortex-M4 Velocidade Nominal: 72Mhz Memoria Flash: 256KB RAM: 64KB Tensão de entrada: 3,3V-6V Tensão de saída: 3,3V	Corrente de saída: 100mA Digital entrada/saída pinos: 34 DAC Resolução: 16 bits PWM saídas: 12 pinos Serial: 3 I2C: 2 UART: 3

3.1.3 Sensores

Em seguida, para obter uma leitura completa dos movimentos de golfe ou quaisquer outros, é necessário recorrer a sensores que sejam capazes de medir a rotação angular e a aceleração nas três dimensões com uma precisão significativa.

Os movimentos no golfe têm maior aceleração e rotação quando comparados com atividades do quotidiano, tais como, andar ou correr. Devido a isso, foi necessário recorrer ao modulo *AltIMu-10 v5* (PTRobotics, 2016a), Figura 3.3, que têm capacidade de medir a rotação angular até ± 2000°/s e aceleração até ± 16 g nos três eixos.

Figura 3.3: AltlMu-10 v5 (PTRobotics, 2016a).

O modulo *AltlMu-10 v5* tem três sensores integrados, descritos na Tabela 3.2. Neste projeto apenas foi utilizado o sensor LSM6DS33, pois não foi possível encontrá-lo separado. O sensor utiliza o protocolo de comunicação I2C sendo as bibliotecas disponibilizadas pelo fornecedor.

Tabela 3.2: AltlMu-10 v5 integrados.

Integrado	Medição
LSM6DS33	Aceleração e rotação nos 3 eixos
LIS3MDL	Magnetómetro nos 3 eixos
LPS25H	Barómetro digital

As principais características dos sensores podem ser observados na Tabela 3.3.

Tabela 3.3: Principais características do AltlMu-10 v5 (PTRobotics, 2016a).

Dimensões: 25 mm × 13 mm × 3 mm Tensão de entrada: 2.5 V-5.5 V Corrente: 5 mA Comunicação: I2C Giroscópio: 16-bit por eixo Acelerómetro: 16-bit por eixo Magnetómetro: 16-bit por eixo Barómetro: 24-bits	Valores de medição: Giroscópio: ±125, ±245, ±500, ±1000, ±2000°/s Acelerómetro: ±2, ±4, ±8, ±16 g Magnetómetro: ±4, ±8, ±12, ±16 gauss Barómetro: 260 mbar to 1260 mbar

3.1.4 Comunicação sem fios

A comunicação entre o telemóvel inteligente e a pulseira é muito importante pois a pulseira não tem nenhum visor ou botões para poder visualizar os resultados ou dar algum comando sendo o telemóvel a única forma de manipular a pulseira e analisar os resultados obtidos pelos sensores. Os dados lidos pelos sensores são sempre enviados para o telemóvel inteligente pois o microprocessador não tem capacidade de armazenamento suficiente. Como tal, o meio de comunicação terá que ter uma velocidade de transmissão e pacotes suficientemente grandes para garantir que os dados chegam tão rápido quanto possível de tempo real.

A comunicação entre a pulseira e o telemóvel inteligente é efetuado via Bluetooth pois é uma interface de baixo custo projetado para ter baixo consumo de energia. É uma tecnologia usada para comunicação entre vários dispositivos electrónicos, sendo muito fácil encontrar documentação para ajudar a entender melhor o seu funcionamento e a forma de a utilizar.

O módulo de comunicação escolhido foi o Bluetooth EDR BEE, Figura 3.4, com uma frequência de trabalho de 2.4Ghz. A distância de transmissão pode ir até aos 30m no espaço livre, o que é suficiente para a aplicação em causa sendo que a taxa de transferência pode ir até 2.1Mbps. Este circuito é alimentado com 3.3V DC e 50mA sendo a sua potência de transmissão de 2.5mW (Dfrobot, 2016).

Figura 3.4: Bluetooth EDR BEE (Dfrobot, 2016).

3.1.5 Fonte de alimentação

Como fonte de energia, recorreu-se a uma bateria de polímero de lítio com uma tensão de 3.7V 850mAh (Figura 3.5) é a melhor opção devido à sua espessura, peso e capacidade energética. A tensão e a potência fornecida por esta bateria é suficiente para alimentar a pulseira durante mais de cinco horas (um dos requisitos do projeto).

Quanto ao modo de carregamento, estas baterias têm um carregamento especial, só podendo ser carregadas com o carregador próprio, Figura 3.6. A bateria tem também embutida proteção contra sobretensão, sobrecarga e tensão mínima (PTRobotics, 2016c).

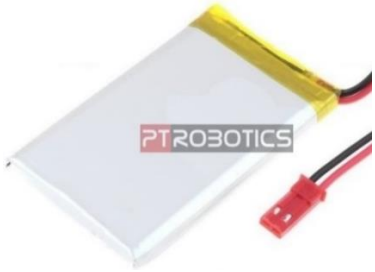


Figura 3.5: Bateria de polímero de lítio 3.7v 850mAh (PTRobotics, 2016c).

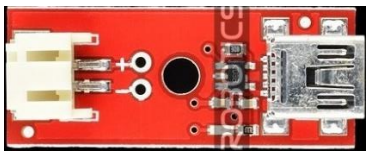


Figura 3.6: Carregador de baterias de polímero de lítio (PTRobotics, 2016b).

3.2 Implementação do Hardware

Após analisar cada componente separadamente, apresentam-se as ligações elétricas, a configuração dos sensores, do Bluetooth e por fim, a programação da pulseira. A Figura 3.7 mostra o esquema das ligações da pulseira, tendo a indicação do nome e do número dos pinos usados para cada componente e as suas ligações com os restantes. A Figura 3.8 mostra a montagem do circuito da pulseira, e por fim, a montagem final no pulso do golfista (ver Figura 3.9).

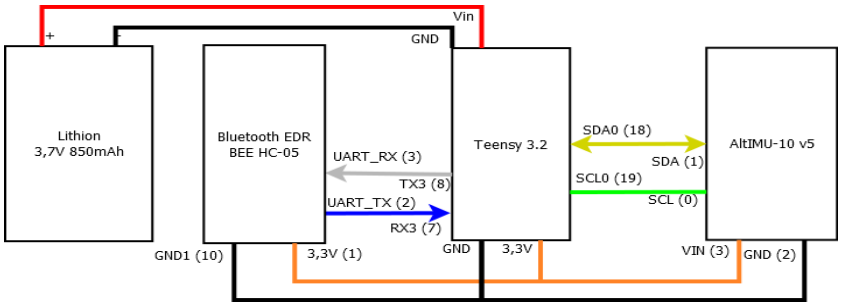


Figura 3.7: Esquema de ligação da pulseira.

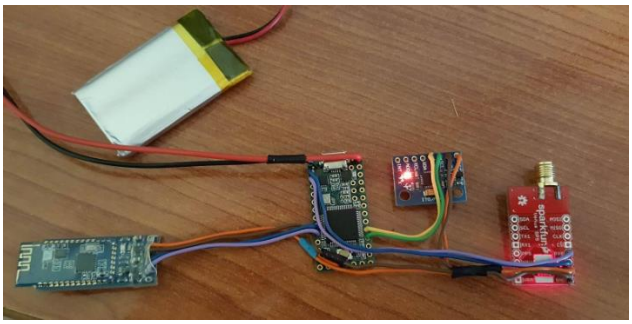


Figura 3.8: Montagem do circuito.

Figura 3.9: Montagem final da pulseira.

3.2.1 Configuração dos Sensores

O módulo *AltlMu-10 v5* vem configurado de fabrico que por exemplo, no caso do acelerómetro e giroscópio, vem configurado para medir até 2g e 200Dps respetivamente (PTRobotics, 2016a). No entanto, estas medidas não são suficientes para medir os movimentos dos golfistas visto haver a necessidade de as medidas serem maiores (por exemplo: 12g e 1800Dps) pelo que há necessidade de fazer a configuração adequada dos mesmos.

Posteriormente, é possível alterar as características do integrado LSM6DS33 (aceleração e rotação do movimento), de duas maneiras. Quanto à primeira é recorrer diretamente à biblioteca do sensor e realizar a alteração do valor do registo "*writeReg(CTRL1_XL, 0x80)*" (Arduino LIbraries, 2016). A segunda, é através da programação, e quando é chamada a função "*imu.writeReg(LSM6::CTRL1_XL, 0x80)*". A seguir é descrito com mais detalhes o significado de cada registo.

O valor LSM6 identifica o integrado LSM6DS33, o sensor que mede a aceleração e a rotação angular, CTRL1_Xl identifica o acelerómetro sendo que no caso do giroscópio o seu valor é CTRL2_G. Por fim, 0x80 é o valor do registo ou valor de configuração que está em formato hexadecimal, sendo convertido de um valor binário onde cada bit ou conjunto de bits é associado a um parâmetro, tanto para o acelerómetro como para o giroscópio. Este valor é composto por 8 bits (Figura 3.10 e Figura 3.11).

ODR_XL3	ODR_XL2	ODR_XL1	ODR_XL0	FS_XL1	FS_XL0	BW_XL1	BW_XL0

Figura 3.10: Valor do registo do CTRL1_X1 (Arduino LIbraries, 2016).

ODR_G3	ODR_G2	ODR_G1	ODR_G0	FS_G1	FS_G0	FS_125	0(1)

Figura 3.11:Valor do registo do CTRL2_G (Arduino LIbraries, 2016).

A seguir, é descrito o significado de cada conjunto de bits para os parâmetros do acelerómetro:

- ODR_XL[3:0] são os bits que correspondem à frequência de trabalho, a configuração da mesma está na Figura 3.12.
- FS_X[1:0] são os dois bits que correspondem à escala da medida de aceleração, por exemplo: 00: ±2 g; 01: ±16 g; 10: ±4 g; 11: ±8 g.
- MW_XL[1:0] são os dois bits que correspondem à largura de banda de frequência do filtro *anti-aliasing*, por exemplo: 00: 400 Hz; 01: 200 Hz; 10: 100 Hz; 11: 50 Hz.

E por fim, no caso dos parâmetros do giroscópio:

- ODR_G [3:0] são os bits que correspondem à frequência de trabalho, Figura 3.12.
- FS_G[1:0] são os dois bits que correspondem à escala da medida da rotação, por exemplo: 00: 245 dps; 01: 500 dps; 10: 1000 dps; 11: 2000 dps.
- FS_125 é o bit responsável pela conversão para 125dps, e é sempre '0' para que haja uma operação adequada do dispositivo.
- (1) é sempre '0'.

ODR_ XL3	ODR_ XL2	ODR_ XL1	ODR_ XL0	ODR selection [Hz] when XL_HM_MODE = 1	ODR selection [Hz] when XL_HM_MODE = 0
0	0	0	0	Power-down	Power-down
0	0	0	1	13 Hz (low power)	13 Hz (high performance)
0	0	1	0	26 Hz (low power)	26 Hz (high performance)
0	0	1	1	52 Hz (low power)	52 Hz (high performance)
0	1	0	0	104 Hz (normal mode)	104 Hz (high performance)
0	1	0	1	208 Hz (normal mode)	208 Hz (high performance)
0	1	1	0	416 Hz (high performance)	416 Hz (high performance)
0	1	1	1	833 Hz (high performance)	833 Hz (high performance)
1	0	0	0	1.66 kHz (high performance)	1.66 kHz (high performance)
1	0	0	1	3.33 kHz (high performance)	3.33 kHz (high performance)
1	0	1	0	6.66 kHz (high performance)	6.66 kHz (high performance)

Figura 3.12: Frequência de Trabalho (Arduino LIbraries, 2016).

Finalmente, o valor escolhido para o registo do acelerómetro foi 0x44, o que significa: 104Hz, 16g e 400Hz, e no caso do giroscópio 0x4C, o que significa:104Hz, 2000dps e 400Hz.

3.2.2 Configuração do Bluetooth

Antes de se usar o Bluetooth para se fazer a comunicação entre a pulseira e o telemóvel inteligente, deve-se efetuar a configuração do mesmo. Deve-se por exemplo mudar o nome, a palavra-chave ou a taxa de transmissão, entre outras. Esta configuração é feita só uma vez, ou sempre que seja necessário reconfigurar a ligação para uma outra aplicação, por exemplo.

Para se fazer a configuração do Bluetooth é necessário seguir os passos:

- Colocar o interruptor "AT Mode" em modo AT (Dfrobot, 2016) deslocando-o para a direita (ver Figura 3.13).
- Conectar o módulo Bluetooth com o Teensy e em seguida ao computador.
- Recorrer ao IDE do Arduíno configurado para o Teensy.
- Ir ao monitor série.
- Configurar o monitor série para "Both NL & CR", "38400".
- Enviar alguns dos comandos AT, que estão descritos na Tabela 3.4.

Figura 3.13: Configurar em modo AT.

Tabela 3.4: Principais comandos AT.

Comando AT	Resposta
AT	Devolve "OK" se a comunicação foi feita com sucesso.
AT+UART?	Devolve a taxa de transferência para qual está configurado.
AT+NOME?	Devolve o nome.
AT+UART=Baud Rate, stop bit, parity bit	Configuração para outros valores. Ex: AT+UART=38400,0,0

3.2.3 Programação da Pulseira

Após a configuração dos sensores e do Bluetooth pode-se então desenvolver e implementar o algoritmo da pulseira, que dará a possibilidade ao utilizador de decidir qual é o sensor que pretende fazer a leitura dos valores e quando fazê-lo, entre outras possibilidades.

O algoritmo implementado na pulseira é composto por duas grandes funções: Setup e Loop.

3.2.3.1 Função Setup

A função Setup é executada após o Teensy ser ligado ou sempre que este seja reiniciado. Tem a principal função de verificar as conexões com os sensores e a comunicação entre o microprocessador e o telemóvel inteligente. Seguidamente são descritas as tarefas que são executadas na função Setup:

- Iniciar a comunicação com o Bluetooth e com os sensores.
- Verificar a comunicação com o Bluetooth e ficar à espera do comando "++" do telemóvel inteligente, este comando podia ser outra coisa qualquer, como por exemplo uma combinação de letras.
- Verificar os sensores e criar as variáveis.

A Figura 3.14 apresenta o fluxograma que mostra a ordem pela qual as tarefas anteriormente descritas são executadas. A taxa de amostragem, do inglês *Sampling Rate*, é também definida na função Setup.

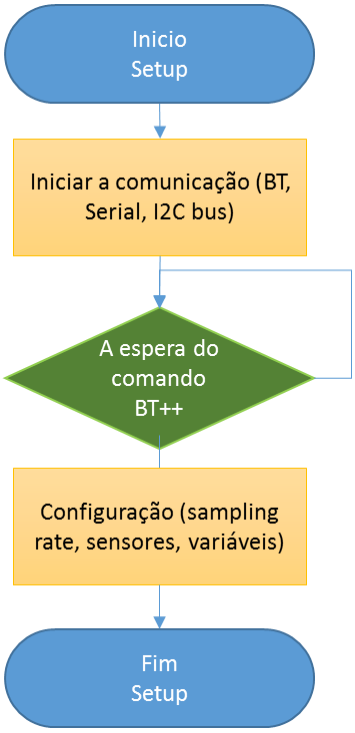


Figura 3.14: Fluxograma da função Setup.

3.2.3.2 Função Loop

A função Loop é chamada depois da função Setup e contém o algoritmo de controlo da pulseira. De seguida serão descritas as suas principais tarefas, Figura 3.15:

- Se não for em modo gravação, imprime as informações do estado dos sensores, isto é, informa quais os sensores que estão ativos para serem lidos e as opções que podemos tomar. Estas opções podem ser o ligar ou desligar a leitura de um dos sensores, ou simplesmente iniciar a gravação dos dados. Posteriormente, executa o comando escolhido e volta ao início do ciclo.
- Se for em modo gravação de dados, lê os valores dos sensores escolhidos para o efeito e envia para o telemóvel. Enquanto não receber o comando "R", cuja função é parar a gravação, o ciclo e as gravações não param.

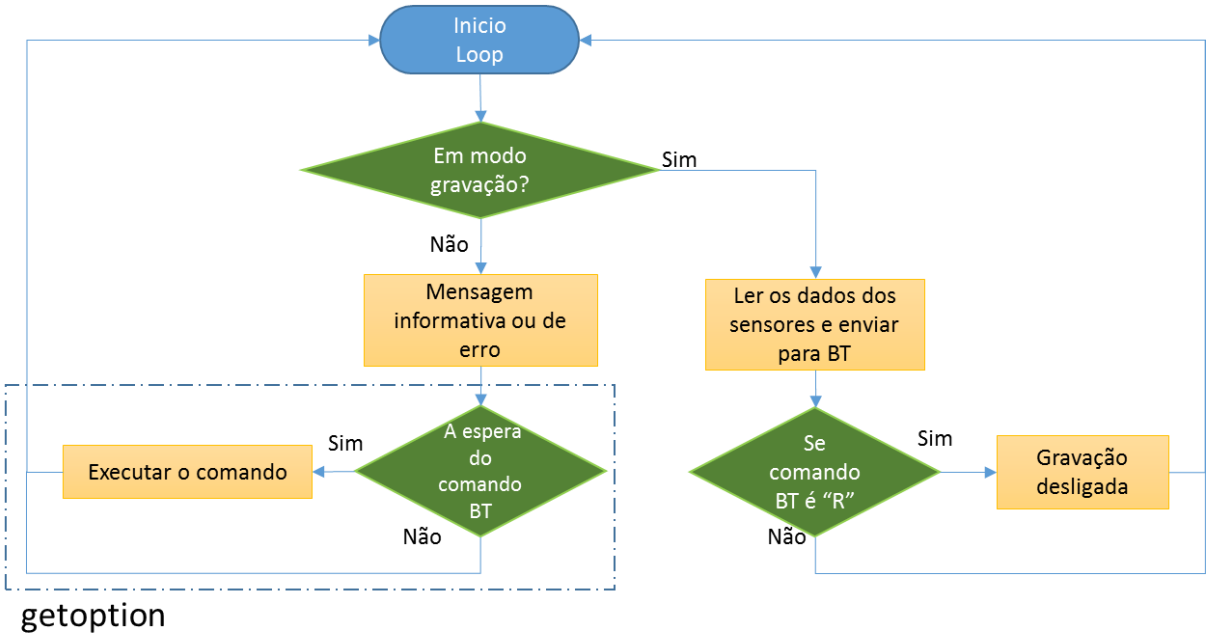


Figura 3.15: Fluxograma da função Loop.

O fluxograma da Figura 3.16 descreve com maior detalhe as condições e os comandos a serem executados na função "getoption". Durante a gravação de dados o telemóvel inteligente envia sempre o comando "S" até que seja enviado outro comando escolhido pelo utilizador, como por exemplo o comando para parar a gravação.

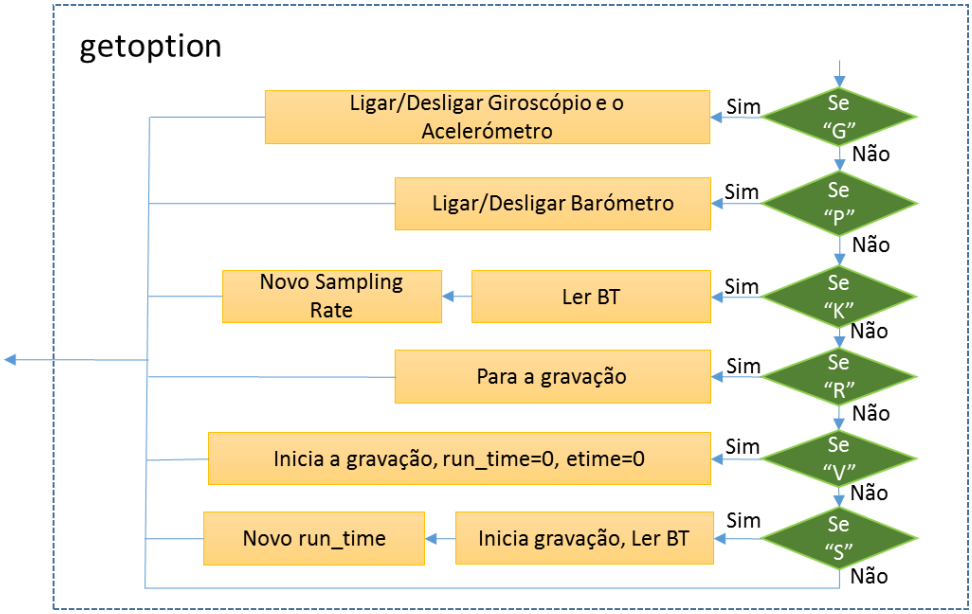


Figura 3.16: Fluxograma da função Getoption.

3.3 Deteção e Identificação de Jogadas

Neste subcapítulo será descrito o algoritmo desenvolvido para detecção de jogadas de golfe, extração dos tempos e os módulos máximos da rotação e aceleração no momento do batimento do taco com a bola e ainda a identificação do tipo de movimento. O algoritmo tem dois grandes blocos (Figura 3.17) onde o bloco "Detecção e Extração" tem como função de verificar se existiu uma jogada ou não. Caso tenha existido, extrai as características, caso contrário volta ao início. O bloco "Classificação" tem como objetivo classificar o tipo do movimento efetuado. O algoritmo só pode ser aplicado depois de obter o sinal completo da gravação pois não é possível verificar uma jogada sem esta ter terminado (Anguita, Ghio, Oneto, Parra, & Reyes-Ortiz, 2012). Para este trabalho de dissertação, os algoritmos foram desenvolvidos e testados no computador, através de *software*. Num projeto futuro poderão ser implementados no telemóvel inteligente de modo que a detecção de jogada, extração de características e identificação do tipo de jogada seja feita de forma automática no próprio telemóvel.

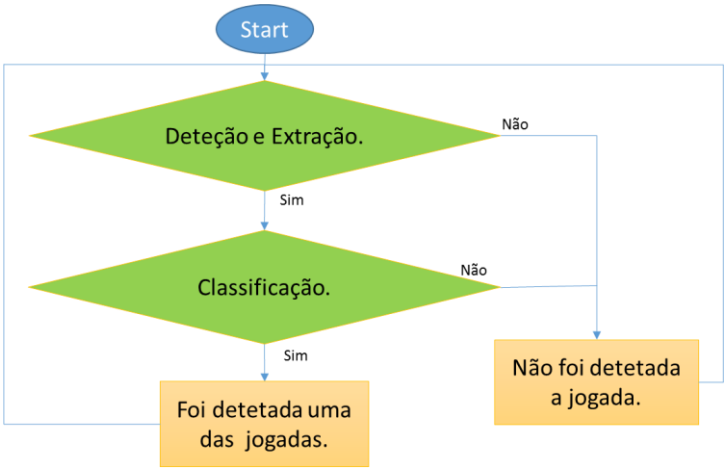


Figura 3.17: Fluxograma do algoritmo desenvolvido.

Neste projeto foi desenvolvido o algoritmo de detecção para as jogadas *Full Swings* e *Pitchings*. A detecção de *Bunker, Chippings* e *Putters* poderá vir a ser implementada num projeto futuro. A Tabela 3.5 mostra os movimentos de golfe a serem estudados nesta tese e os tacos de referência utilizados para o ensaio.

Para o estudo do algoritmo foram efetuadas em média quatro gravações para cada tipo de movimento tendo, em alguns casos, sido repetidas. Algumas repetições foram efetuadas em datas e campos diferentes com seis golfistas diferentes.

Tabela 3.5: Os movimentos de golfe para o estudo.

Swings (*Clubs*/Tacos)	*Full swing* *(Short Iron)*	*Full swing* *(Long Iron)*	*Full swing* *(Driver)*	*Pitching* (60 metros)	*Pitching* (30 metros)
A Referência utilizada para o ensaio	PW	7i	*Driver*	PW	SW

A Tabela 3.6 mostra o *handicap*, o género e a idade dos golfistas (os nomes dos golfistas foram omitidos para proteção de identidade). Durante as gravações dos movimentos de golfe com a pulseira foram captadas algumas fotografias dos golfistas (ver Figura 3.18).

Figura 3.18: Fotografias captadas durante as gravações de movimentos de golfe.

Tabela 3.6: Os golfistas para o estudo.

Nome	HCP	Género	Idade
Jogador 1	0	Masculino	60
Jogador 2	16,6	Masculino	41
Jogador 3	17	Masculino	37
Jogador 4	0.8	Masculino	35
Jogador 5	11	Masculino	32
Jogador 6	26	Masculino	43

3.3.1 Deteção de Jogadas

Antes da extração de características de uma jogada e da classificação do tipo de jogada é necessário haver uma detecção da jogada, isto é, tem de ser realizado um dos movimentos de golfe para se poder avançar com a extração e classificação.

Primeiro é necessário visualizar no tempo cada parte do movimento para um movimento completo de golfe e entender o seu significado.

A Figura 3.19 mostra o gráfico do giroscópio com os três eixos sendo o módulo do movimento avaliado pela pulseira calculado de acordo com a equação (3.3.1).

$$Módulo(x, y, z) = \sqrt{x^2 + y^2 + z^2} \qquad (3.3.1)$$

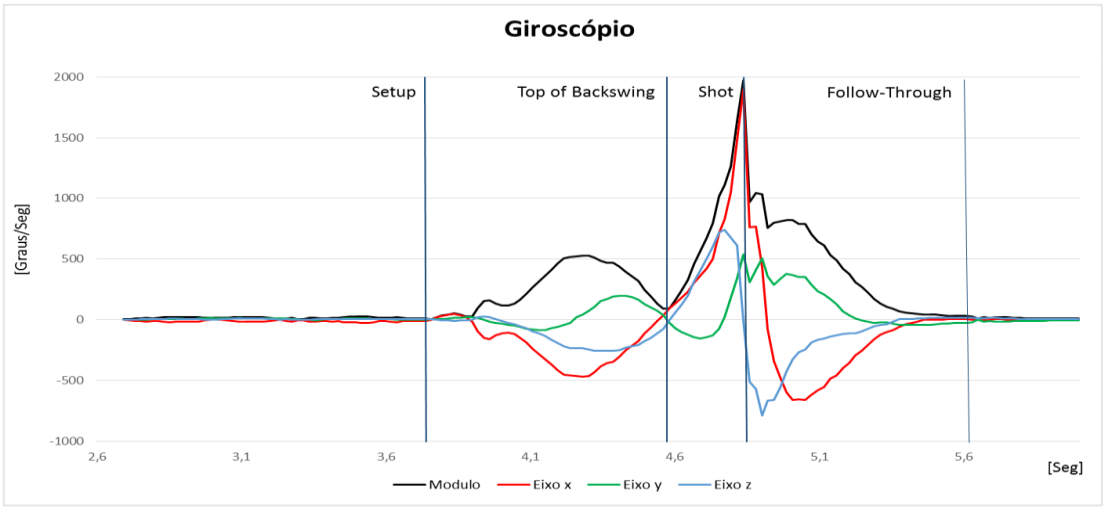


Figura 3.19: Os instantes de um movimento completo de golfe.

Seguidamente são descritos os termos utilizados, em inglês, para caracterizar um movimento completo do golfe:

- *Setup* é o momento do fim da preparação e o início do movimento para trás.
- *Top of Backswing* é o momento em que o taco se encontra em cima, isto é, o momento antes de se iniciar o movimento para baixo.
- *Shot* é o momento quando o taco de golfe toca na bola.
- *Follow-Through* é o momento de acabamento, ou de repouso.

Ao conseguir localizar os instantes do movimento, facilmente se conseguem calcular os tempos descritos na Tabela 2.1, onde a diferença entre *Top of Backswing* e *Setup* nos dará o *Backswing Time*, a diferença entre *Shot* e *Top of Backswing* é denominada *Downswing Time*, e por fim, a diferença entre *Follow-Through* e *Shot* é o *Follow-Through Time*.

Posteriormente é necessário entender a diferença entre os vários tipos de jogada. Como mencionado na Tabela 2.2, os *Full swings (Long Iron, Short Iron e Drive)* são todos movimentos completos de golfe, porém para cada um deles são utilizados tacos diferentes. No entanto, verifica-se que os gráficos gerados pelos sensores são idênticos, mas diferentes de *Pitching* de 30 e 60 metros. A Figura 3.20 mostra os módulos sobrepostos de *Full Swing* e *Pitchings*. Pode-se observar que em termos de comportamento, o sinal é muito idêntico, mas em termos de valores absolutos são diferentes, isto é, quanto menor é o movimento, menores são os tempos e o módulo de rotação angular.

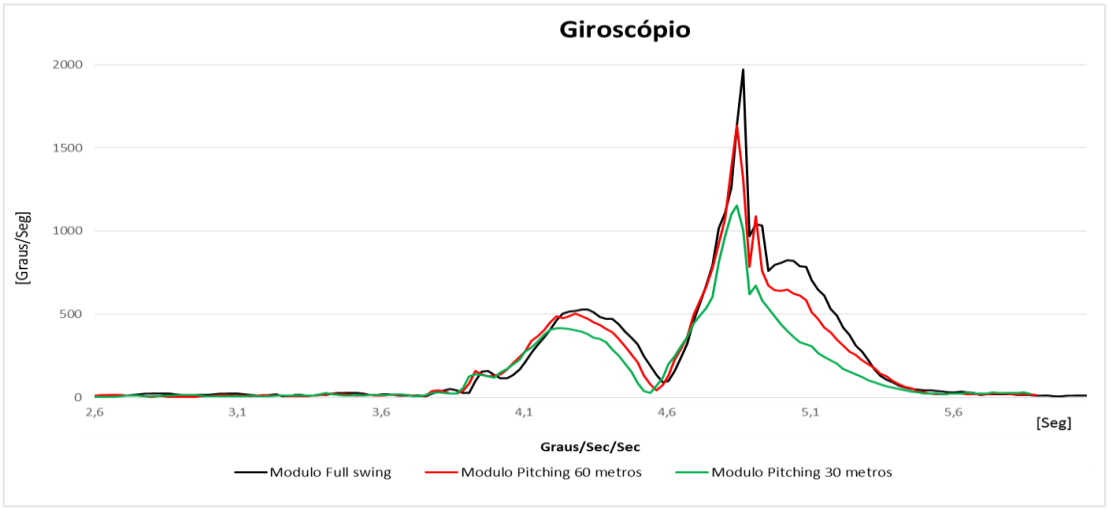


Figura 3.20: Sobreposição dos módulos do *Full swing*, *Pitching* de 60 e 30 metros.

Analisando os vários gráficos dos diferentes golfistas para o mesmo tipo de movimento observou-se que cada jogador tem uma rotação diferente, devido à posição da pulseira ou mesmo devido à sua forma de agarrar no taco. Apesar disto, existem algumas características que todos têm em comum, embora com pequenas variações. A Figura 3.21 mostra o gráfico do giroscópio em que se assinalam com elipses cor de rosa os pontos em que os sinais dos diversos golfista coincidem entre si. Fora destes intervalos, o comportamento dos sinais pode ser diferente ou não. A detecção destes pontos pode servir para garantir que é detetado um movimento do golfe e não outro movimento qualquer, como por exemplo, o simples abanar do braço.

A seguir são descritos com mais detalhe os pontos em comum que os jogadores têm durante a execução de um movimento de golfe, tanto para *Full swings* com para *Pitchings*:

- P1, os eixos X e Z estão próximo de zero antes de iniciar o *Backswing* e tendem para negativo depois.
- P2, no momento quando há mudança de *Backswing* para *Downswing* os eixos X e Z passam de negativo para positivo, no caso do eixo Y é ao contrário, positivo para negativo.
- P3, o eixo Z tem um máximo nos instantes antes do P4 e os valores dependem dele.
- P4, o eixo X tem um valor muito próximo do módulo máximo da rotação no momento da batida da bola.
- P5, o eixo Y tem um máximo um pouco depois do P4 e os valores dependem dele.
- P6, o eixo Z tem um mínimo um pouco depois do P4 e os valores dependem dele.
- P7, os eixos X e Z estão com valores negativos antes de *Follow-Through* e próximo de zero depois.

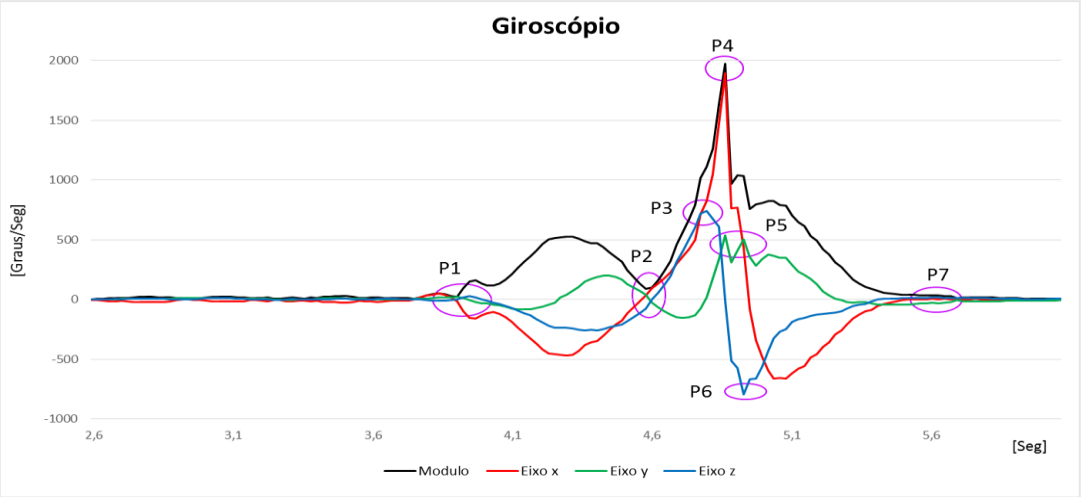


Figura 3.21: Os pontos em comum.

3.3.2 Extração de Características

É importante conhecer os tempos de cada parte de um movimento de golfe e também a rotação e aceleração no momento do batimento da bola, pois são estes os fatores que influenciam o resultado final da mesma. A Figura 3.22 mostra a sobreposição do módulo do giroscópio e do acelerómetro e podendo-se observar que no momento do impacto do taco de golfe com a bola, este tem a aceleração e a rotação máxima.

Para se poder visualizar o módulo da aceleração na mesma escala com o módulo de rotação, foi necessário multiplicar a aceleração por 100, normalizada a escala. Ao estudar vários gráficos notou-se que, por vezes, o máximo de rotação e aceleração não se verificam exatamente no mesmo instante podendo variar até três amostras. Isto deve-se principalmente a um atraso na leitura dos diferentes sensores.

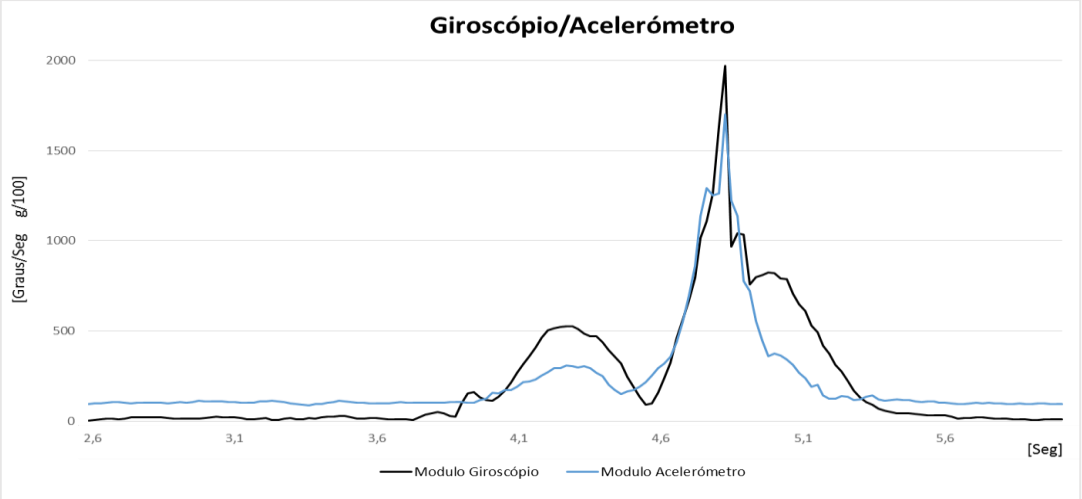


Figura 3.22: Sobreposição do módulo do Giroscópio com o do Acelerómetro.

Depois de se descrever e definir os pontos importantes dos gráficos gerados pelos sensores, como os tempos de cada movimento para um movimento completo de golfe e o módulo máximo da rotação angular e a aceleração no momento da batimento da bola, foi então necessário extrair estes valores manualmente, para se poder chegar a um algoritmo que permita executar esta tarefa automaticamente.

A Tabela 3.7 mostra os resultados das médias e dos desvios padrão calculados manualmente a partir de todas a amostras que foram gravadas para o estudo do algoritmo de detecção e identificação de jogadas.

Como era espectável, as médias dos *Full swings* (*Long Iron*, *Short Iron* e *Drive*) são muito próximas, dificilmente se conseguindo distinguir visualmente os gráficos gerados pelos sensores pelo que para a criação do algoritmo se considerou um único movimento completo.

Ao extrair as características manualmente foram seguidas algumas condições, como os máximos dos módulos que foram considerados como valores de referência.

Ao encontrar estes valores é necessário andar para trás e para a frente no tempo e impôr diversas condições para extrair os tempos da cada parte do movimento. Esses tempos são tirados a partir do módulo da rotação

Tabela 3.7: As médias e os desvios padrão dos máximos dos módulos e dos tempos.

Movimento		Max. Mod. Aceleração (g)	Max. Mod. Rotação (°/s)	*Backswing Time* (s)	*Downswing Time* (s)	*Follow-Through Time*
Full Swing (Short Iron)	Média	16,86	1861,90	1,01	0,25	1,02
	σ	3,30	157,25	0,23	0,03	0,29
Full Swing (Long Iron)	Média	16,98	1822,61	1,01	0,25	0,97
	σ	3,41	182,30	0,04	0,04	0,32
Full Swing (Driver)	Média	17,54	1877,64	1,01	0,26	0,84
	σ	2,50	155,66	0,20	0,04	0,22
Pitching (60 metros)	Média	12,07	1532,20	0,92	0,28	0,94
	σ	2,91	217,49	0,17	0,04	0,31
Pitching (30 metros)	Média	9,88	1121,12	0,79	0,30	0,76
	σ	2,78	166,49	0,09	0,05	0,21

A Tabela 3.8 mostra as condições para detecção dos tempos e dos módulos máximos para um movimento completo de golfe que nesta fase são aplicadas ao módulo do giroscópio. O parâmetro S*hot Time* não está mencionado, mas é conhecido, pois encontra-se no mesmo instante do máximo do módulo da rotação.

Tabela 3.8: Condições para extração dos máximos e dos tempos.

Movimento	Max. Mod. Rotação (°/s)	Max. Mod. Aceleração (g)	*Setup Time* (°/s)	*Top of Backswing Time* (°/s)	*Follow-Through Time* (°/s)
Full Swing	1250 >	8>	16 <	70 <	30 <
Pitching (60metros)	1150 >	7 >	16 <	70<	30 <
Pitching (30metros)	840< e ≤1150	5 < e ≤ 17	16 <	45 <	30 <

Ter as condições da Tabela 3.8 não é suficiente, pois estas condições têm que estar localizadas num determinado intervalo de tempo e não apenas em amplitude. Caso não se encontrem dentro desses parâmetros, não é considerado um movimento válido de golfe. A Tabela 3.9 mostra o intervalo de tempo em que os tempos se podem encontrar.

Estes valores foram tirados do estudo das amostras todas utilizadas para o desenvolvimento deste algoritmo. Estes intervalos estão definidos relativamente ao ponto de referência que é o máximo do módulo da rotação. Com base em todas estas condições, é possível garantir a extração de características com a maior precisão e uma detecção de jogada correta.

Tabela 3.9: Localização dos tempos.

Movimento	Setup Time (s)	*Top of Backswing Time* (s)	*Follow-Through Time* (s)
Full Swing	1,254≤ e ≤1,914	0,220≤ e ≤0,440	0,506≤ e ≤1,628
Pitching (60metros)	1,254≤ e ≤1,914	0,220≤ e ≤0,480	0,500≤ e ≤1,626
Pitching (30metros)	1,144≤ e ≤1,496	0,220≤ e ≤0,506	0,440≤ e ≤1.210

Tendo as condições de detecção de jogada e extração de características pode-se então criar o algoritmo automatizado. Na Figura 3.23 mostra-se o fluxograma do algoritmo para o bloco "Detecção e Extração". A detecção de jogadas e extração de características é executada na mesma altura para se poupar o número de operações.

Seguidamente é descrito com mais profundidade o fluxograma da Figura 3.23. A Tabela 3.10 e a Tabela 3.9 mostram com mais detalhe cada verificação feita pelo algoritmo. Como se pode constatar, os valores dependem muito do módulo máximo da rotação havendo sempre um intervalo onde os eixos se podem encontrar pelo que o mesmo golfista pode não obter sempre o mesmo resultado. Este facto levou a que se tivesse de introduzir uma tolerância.

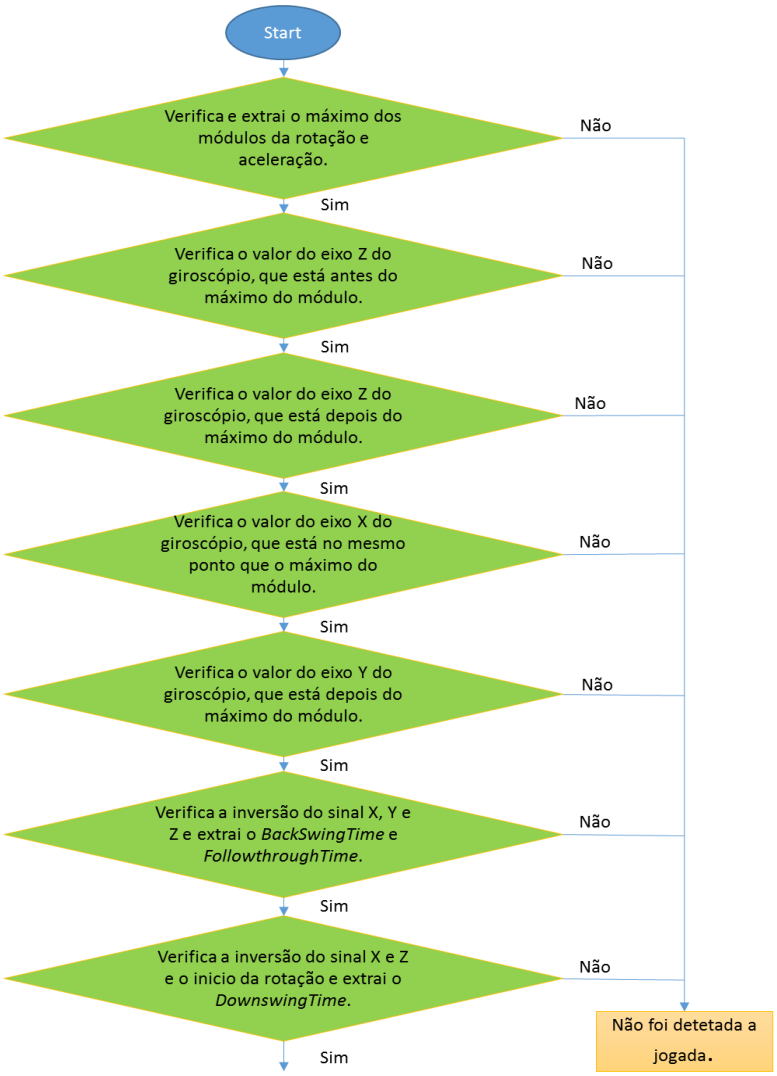


Figura 3.23: Fluxograma do bloco "Detecção e Extração".

Tabela 3.10: Detalhes das verificações.

Verificação	Detalhes
Verifica e extrai o máximo dos módulos da rotação e aceleração.	As condições estão na Tabela 3.8 e Tabela 3.9.
Verifica o valor do eixo Z do giroscópio, que está antes dos máximos do módulo da rotação.	0,3*Max.Mod.Rotação < Eixo Z da rotação < 0,5* Max.Mod.Rotação. Entre 0.066s e 0.132s antes do ponto Max.Mod.Rotação
Verifica o valor do eixo Z do giroscópio, que está depois dos máximos do módulo da rotação.	-0.4*Max.Mod.Rotação < Eixo Z da rotação < -0.1* Max.Mod.Rotação. Entre 0.022s e 0.132s depois do ponto Max.Mod.Rotação.

Verifica o valor do eixo X do giroscópio, que está no mesmo ponto que o módulo máximo da rotação.	0.8*Max.Mod.Rotação < Eixo X da rotação < 0.99* Max.Mod.Rotação No mesmo tempo que Max.Mod.Rotação.
Verifica o valor do eixo Y do giroscópio, que está depois do máximo do módulo.	0.05*Max.Mod.Rotação < Eixo Y da rotação < 0.4* da Rotação; Entre 0.022s e 0.132s depois do ponto Max.Mod.Rotação.
Verifica a inversão do sinal X, Y e Z e extrai o *Backswing Time* e *Follow-Through Time.*	Eixo Z e X da rotação está negativo antes de *Top of Backswing Time* 0,11s e 0,11 depois positivo. Eixo Y é ao contrário, primeiro positivo e depois negativo. Os valores para extrair dos tempos estão na Tabela 3.8 e Tabela 3.9.
Verifica a inversão do sinal X e Z e o inicia da rotação e extrai o *Downswing Time.*	Depois de 0.22s de se detetar o Setup os eixos X e Z da rotação estão com valores negativos.

3.3.3 Classificação

A diferenciação entre cada tipo de movimento de golfe reflete-se nos tempos, na rotação e na aceleração máxima no momento do batimento da bola. Para isto é necessário utilizar um algoritmo de classificação para fazer a distinção entre os tipos de movimentos de golfe. Primeiro é necessário extrair estas características, obtidas a partir de todas as gravações. Depois, testar os diversos classificadores para se escolher o melhor. Por fim, ter à saída umas das classes dos tipos das jogadas, que anteriormente foram atribuídas (Bunkheila, 2016) Figura 3.24.

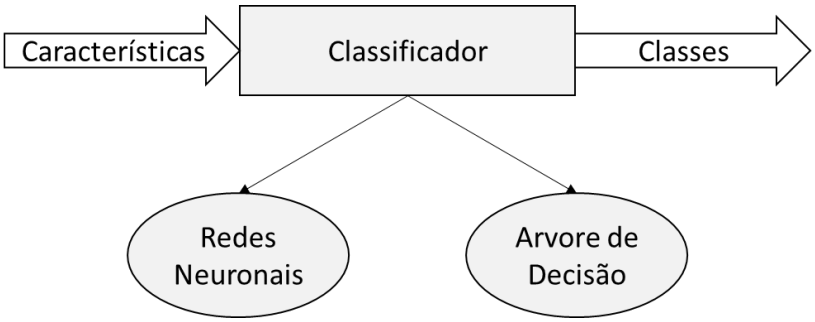


Figura 3.24: Fluxograma da Classificação.

Para avaliar a qualidade do classificador a ser escolhido para classificar os tipos de movimento de golfe, é utilizada a equação (Fonseca, 1994) (3.3.2), onde é dividido o número de casos errados pelo número total de casos testados. Quanto mais baixo for o resultado, melhor é o classificador.

$$Percentagem\ de\ erro = \frac{Número\ de\ erros}{Número\ de\ casos\ testados} \quad (3.3.2)$$

As características extraídas das gravações das jogadas estão na Tabela 3.11 sendo utilizadas para se treinar os classificadores.

Tabela 3.11: Características das Jogadas.

Características
Módulo máximo da rotação.
Módulo máximo da aceleração.
Backswing Time
Downswing Time
Follow-Through Time

É necessário dar uma classe a cada tipo de movimento de golfe e o total do número de gravações para cada classe de modo a ter a possibilidade de calcular a qualidade do classificador. A diferenciação entre os *Full swings (Short Iron, Long Iron* e *Drive*) não é feita, pois é difícil distingir o mesmo movimento utilizando tacos de golfe diferentes. É então atribuído a todos os *Full swings* a mesma classe, neste caso classe 1. Ao *Pitching* de 60 metros é atribuída a classe 2 e ao de 30 metros a classe 3, como mostra a Tabela 3.12. Os golfistas com os quais foram feitas as gravações estão mencionados na Tabela 3.6.

Tabela 3.12: Classes das Jogadas.

Movimento	Classe	Total
Full swing (Short Iron) *Full swing (Long Iron)* *Full swing (Driver)*	1	99
Pitching (60 metros)	2	33
Pitching (30 metros)	3	31

3.3.3.1 Árvore de Decisão

Os algoritmos baseados em árvores de decisão podem ser aplicados com diversos tipos de dados podendo a estrutura final do classificador ser facilmente manipulada e compreendida pelo que encontra muitas aplicações na área de medicina, finanças, agricultura, engenharia e outras, onde apoia a tomada de decisão. Áreas como a avaliação de efeitos secundários no consumo de alguns medicamentos, a aprovação de crédito, a classificação de doenças de plantas e os sistemas de controlo inteligente fazem também um uso extensivo deste tipo de classificadores (Fonseca, 1994).

A árvore de decisão utiliza uma estratégia de dividir para conquistar, onde um problema complexo é decomposto em subproblemas mais pequenos. Esta estratégia é recursivamente aplicada a cada subproblema, até que se encontra uma solução para cada um dos problemas mais simples. Cada ramo da árvore tem um conjunto de condições até que se encontra uma folha que contêm a classe a atribuir. A desvantagem deste tipo de classificadores é a possibilidade de criação de um grande número de ramos por vezes desnecessários (Fonseca, 1994).

Para se criar o algoritmo de classificação baseado numa árvore de decisão foi usado o *software CART* da Salford Systems [16](Salford Systems, 2016), que tem uma licença grátis para trinta dias disponibilizada online. A Tabela 3.13 mostra uma pequena amostra da base de dados utilizada para se treinar a árvore de decisão. Foi selecionada a classe como "Target" e as características como "Predictor", e por fim, utilizou-se o critério de Gini.

Tabela 3.13: Pequena amostra de base de dados.

Max.Mod. Aceleração (g)	Max.Mod. Rotação (°/ s)	*Backswing Time* (s)	*Downswing Time* (s)	*Follow-Through Time* (s)	Classe
13,82	1651,916	1,003	0,24	1,36	1
14,316	1683,93	1,025	0,22	1,277	1
14,304	1951,316	1,441	0,241	1,135	2
14,223	1661,356	0,958	0,241	1,134	2
15,093	1935,305	0,897	0,236	0,609	3
….	….	….	….	….	….

3.3.3.2 Rede Neuronal

As redes neuronais são modelos que permitem implementar modelos de processos, controladores, sistemas de decisão, conhecimentos de padrões que se assemelha a um modelo simplificado do cérebro humano, onde o conhecimento é adquirido do meio envolvente através de um processo de aprendizagem e é guardado nas interligações entre os neurónios (Santos, 2016). O objetivo das redes neuronais é ser mais eficaz em pequenas tarefas específicas, no que diz respeito a tomadas de decisão, do que o cérebro humano, onde na presença de um problema, faz reconhecimento automático deste e ativa a saída correspondente à classe a que lhe pertence. São três os parâmetros principais que determinam a arquitetura de uma rede neuronal: o número de neurónios da camada de entrada que é igual à quantidade de atributos que deve ser usada para se alimentar a rede; o número de neurónios da camada interna e os neurónios da camada de saída, que é normalmente determinada pela quantidade de classes que queremos classificar, como mostra a Figura 3.25, (Fonseca, 1994).

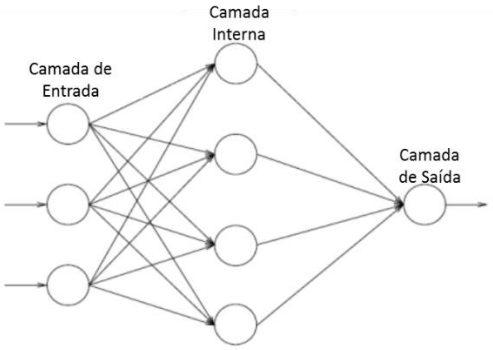


Figura 3.25: Exemplo de uma rede neuronal (Fonseca, 1994).

Ao aumentar o número de neurónios na camada interna, a rede consegue normalmente melhorar a resolução de problemas mais complexos, tendo como principal desvantagem o aumento do tempo do treino da rede (Santos, 2016).

Para se criar e treinar a rede neuronal utilizou-se a aplicação "Neural Net Pattern Recognition" disponibilizada pelo programa MatLab, na versão R2015a. Na Tabela 3.14 está um exemplo dos dados a treinar, onde as características de cada amostra estão organizadas em colunas mostrando-se na Tabela 3.15 a classe correspondente. Primeiramente é necessário carregar os dados do treino e da classe anteriormente preenchidos, seguidamente definir o número de camadas, e por fim, treinar as redes neuronais.

Tabela 3.14: Exemplo de dados para treino das redes neuronais.

13,82	14,32	14,30	14,22	15,09	22,12	…
1651,92	1683,93	1951,32	1661,36	1935,31	2038,39	…
1,00	1,03	1,44	0,96	0,89	0,92	…
0,24	0,22	0,24	0,24	0,24	0,26	…
1,36	1,28	1,14	1,13	0,61	1,68	…

Tabela 3.15: Exemplo de classificação para os dados.

1	1	0	0	0	0	…
0	0	1	1	0	0	…
0	0	0	0	1	1	…

A Figura 3.26 mostra a arquitetura da rede neuronal configurada para se treinar as redes neuronais.

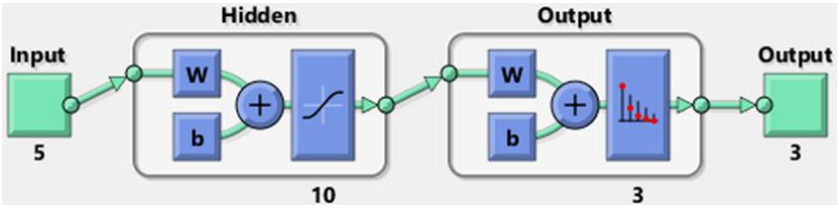


Figura 3.26: Arquitetura da Rede Neuronal.

4 Resultados Experimentais

Uma vez realizado o estudo relativo ao âmbito desta dissertação é necessário observar os resultados experimentais por forma a poder aperfeiçoá-los.

Numa primeira fase foi feita uma análise dos resultados da árvore de decisão e da rede neuronal utilizando as gravações dos movimentos de golfe utilizadas para o estudo do algoritmo de detecção e identificação de jogadas.

A seguir passou-se à fase de testar os algoritmos de classificação com as novas gravações e com novos golfistas.

Por fim, são tiradas as conclusões de todas as experiências efetuadas anteriormente.

4.1 Resultados da Árvore de Decisão

Foram criadas duas arquiteturas baseadas em árvores de decisão, com o objetivo de abordar dois métodos diferentes para o mesmo problema. Na primeira a árvore de decisão é criada a partir das todas as classes, enquanto na segunda se separam as classes. Para se testarem os algoritmos, foi necessário converter a árvore de decisão criada pelo programa Salford para a linguagem C e também para Excel para se poderem testar de imediato. Primeiro os algoritmos são testados com a base de dados com base na qual a árvore foi gerada, sendo a seguir testados com dados de novos golfistas (Du, Du, Zhan, & Zhan, 2002).

4.1.1 Árvore de Decisão com todas as classes

Após ser desenvolvido o algoritmo de classificação com a árvore de decisão com todas as classes é testado com a base de dados para a qual foi criada, de modo a calcular a qualidade do mesmo.

A Tabela 4.1 mostra os resultados da matriz da confusão da árvore de decisão com todas as classes, para classe 1 (*Full swing*). Dos 99 cenários possíveis, 97 amostras foram classificadas como classe 1 e 2 como classe 2 (*Pitching* 60 metros). No caso da classe 2, dos 33 cenários possíveis, 32 amostras foram classificadas como classe 2 e 1 amostra como classe 1. Por fim, no caso da classe 3 (*Pitching* 30 metros), as 31 amostras foram classificadas como classe 3. Isto significa que a classe 1 pode ser confundida com a classe 2, a classe 2 pode ser confundida com a 1, por fim, a classe 3 não é confundida. Recorrendo à equação (3.3.2) a percentagem de erro do classificador é de 1,84%.

Tabela 4.1: Matriz da confusão da árvore de decisão com todas as classes.

Classe atribuída pelo classificador	Classe real		
	1	2	3
1	97	1	0
2	2	32	0
3	0	0	31

4.1.2 Árvore de Decisão hierárquica com classes separadas

A árvore de decisão hierárquica é desenvolvida de seguinte forma: primeiro é feita a divisão entre *Full swings* e *Pitchings* e de seguida os *Pitchings* são divididos em *Pitching* de 60 e 30 metros. Na primeira divisão é atribuída a classe 1 aos Full Swings e a classe 2 aos Pitchings. Na segunda divisão, é atribuída a classe 2 aos Pitching de 60 metros e a classe 3 aos Pitching de 30 metros. A Tabela 4.2 mostra a matriz de confusão da primeira divisão. Para a classe 1 (*Full swings*), os 99 cenários possíveis são classificados como classe 1. No caso da classe 2 (*Pitchings*), dos 64 cenários possíveis, 62 amostras foram classificadas como classe 2 e 2 amostras como classe 1. Resumindo, nesta primeira divisão, a apenas duas amostras não foi atribuída a classe correta.

Tabela 4.2: Matriz da confusão da Full Swing e Pitching.

Classe atribuída pelo classificador	Classe real	
	1	2
1	99	2
2	0	62

Depois é criado outro algoritmo para a segunda divisão dos *Pitchings* que foram classificados corretamente, isto é, desenvolveu-se a árvore de decisão da segunda divisão utilizando as 62 amostras classificados corretamente da primeira divisão e não os 64 que existiam inicialmente. A Figura 4.1 mostra as divisões com mais detalhe e os resultados dos testes.

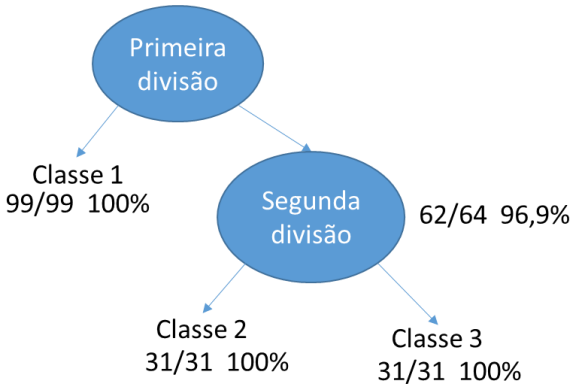


Figura 4.1: Os resultados da árvore de decisão hierárquica com classes separadas.

Finalmente, é feita a união das duas partes dos algoritmos e a mesma é testada. A Tabela 4.3 mostra a matriz de confusão da união, e como era de esperar, a classe 1 é classificada corretamente. No caso da classe 2, apenas duas amostras são classificadas como classe 1, e por fim, são corretamente classificados todos os exemplos da classe 3. Pode assim concluir-se que na primeira divisão apenas duas amostras da classe 2 não foram classificadas corretamente enquanto a segunda divisão obteve 100% de sucesso obtendo-se assim uma percentagem de erro geral do classificador de 1,23%.

Tabela 4.3: Matriz da confusão da árvore de decisão hierárquica com classes separadas.

Classe atribuída pelo classificador	Classe real		
	1	2	3
1	99	2	0
2	0	31	0
3	0	0	31

4.1.3 Testes com novos Jogadores utilizando a Árvore de Decisão

Testando os classificadores com a base de dados com a qual eles foram desenvolvidos obtêm-se percentagens de erro de 1,84% para a árvore de decisão com todas as classes e de 1,23% para a árvore de decisão hierárquica. No entanto, estes resultados não garantem que se possam obter os mesmos resultados com novos dados de outros golfistas.

Surge assim a necessidade de testar os algoritmos de classificação com novos jogadores e avaliar qual é a eficiência do classificador nos casos diferentes daqueles com base nos quais foi desenvolvido.

A Tabela 4.4 mostra os novos golfistas escolhidos para que as gravações dos seus movimentos de golfe sejam testados com os algoritmos de classificação com árvore de decisão. Para isto, foram gravadas 4 jogadas para cada tipo de *Swing*, o que dá 80 gravações, 48 *Full Swings*, 16 *Pitchings* de 60 metros e 16 *Pitchings* de 30 metros.

Tabela 4.4: Os novos jogadores para testes experimentais.

Nome	HCP	Género	Idade
Jogador Novo 1	19,5	Masculino	45 Anos
Jogador Novo 2	16	Masculino	64 Anos
Jogador Novo 3	10	Masculino	71 Anos
Jogador Novo 4	17	Masculino	55 Anos

4.1.3.1 Testes com os novos jogadores utilizando a árvore de decisão com todas as classes

Os resultados do algoritmo com a árvore de decisão com todas as classes são apresentados na Tabela 4.5. Para a classe 1 (*Full swing*), dos 48 cenários possíveis, 29 amostras foram classificadas como classe 1, 14 como classe 2 (*Pitching* 60 metros) e 5 como classe 3 (*Pitching* 30 metros). No caso da classe 2, dos 16 cenários possíveis, 8 amostras foram classificadas como classe 2, 1 amostra como classe 1 e 7 como classe 3. Por fim, no caso da classe 3, dos 16 cenários possíveis, 13 amostras foram classificadas como classe 3, 2 amostra como classe 2 e 1 como classe 1. Com isto, se pode concluir que a classe 1 pode ser confundida facilmente com a classe 2, a classe 2 ser confunda com 3, e por fim, a classe 3 é a menos confundida. A percentagem de erro do classificador com árvore de decisão com todas as classes com os novos jogadores é de 37,5%.

Tabela 4.5: Matriz da confusão com os novos jogadores utilizando a árvore de decisão.

Classe atribuída pelo classificador	Classe real		
	1	2	3
1	29	1	1
2	14	8	2
3	5	7	13

4.1.3.2 Testes com os novos jogadores utilizando a árvore de decisão hierárquica

Os resultados do algoritmo com a árvore de decisão hierárquica são mostrados na Tabela 4.6. São classificadas corretamente 17 amostras em 48 possíveis para classe 1, os restantes, 22 amostras são classificadas como classe 2 e 9 amostras como classe 3. No caso da classe 2, foram acertados 7 em 16, os não acertados foram para classe 3. Por fim, são acertados 14 em 16 para classe 3, os restantes, foram classificados como classe 2 e 3. A classe 1 é facilmente confundida com a classe 2 e 3, a classe 2 com a classe 3, sendo a classe 3 a menos confundida. A percentagem de erro do classificador com árvore de decisão hierárquica com classes separadas com os novos jogadores é de 52,5%.

Tabela 4.6: Matriz da confusão com os novos jogadores utilizando a árvore de decisão hierárquica.

Classe atribuída pelo classificador	Classe real		
	1	2	3
1	17	0	1
2	22	7	1
3	9	9	14

4.1.4 Comparação de resultados

Ao testar as amostras que foram utlizadas para criar a árvore de decisão com todas as classes e a árvore hierárquica com classes separadas obtiveram-se bons resultados (erro de 1,84% e 1,23%, respetivamente) havendo portanto pouca diferença entre eles. No caso dos novos jogadores, no entanto, obtiveram-se piores resultados (37,5% e 52,5%, respetivamente) sendo a árvore de decisão com todas as classes significativamente melhor.

Os maus resultados da classificação dos movimentos de golfe dos novos jogadores devem-se principalmente ao facto dos jogadores serem de uma faixas etárias muito diferentes de aqueles com base nos quais as árvores foram criadas. Uma pessoa na casa dos 40 anos tem maior força e melhores tempos se comparar o mesmo movimento com uma pessoa na casa dos 60. Por isso, o classificador classifica alguns *Full Swings* como *Pitching* de 60 metros, e *Pitching* de 60 metros como *Pitching* de 30 metros, devidas as suas capacidades físicas de cada jogador.

4.2 Resultados da Rede Neuronal

Para se treinar a rede neuronal com todas as classes foi usada a mesma base de dados que foi usada para se criar a árvore de decisão. Simular o algoritmo de classificação com redes neuronais é mais fácil ao utilizar o *software* MatLab, pois basta chamar a função "*figure, plotconfusion(t,net(x))*", onde na variável "*x*" se passa a base de dados para qual é necessário fazer os testes, e na variável "*t*" as respetivas classes. No entanto, passar o algoritmo resultante para uma outra programação, como por exemplo, a linguagem C, é muito difícil devido à sua complexidade.

Foi necessário treinar a rede neuronal várias vezes até se chegar ao melhor resultado, tendo-se obtido a matriz de confusão descrita na Figura 4.2. Como se pode constatar, a rede acertou 97 em 99 amostras para classe 1, tendo aos restantes exemplos sido atribuída a classe 2. No caso da classe 2, a rede acertou 16 exemplos em 33 tendo os 8 dos restantes exemplos sido classificados como classe 1 e 9 como classe 3. Por fim, no caso da classe 3, foram corretamente classificados 30 exemplos em 31 tendo ao restante sido atribuída a classe 2. Pode-se assim concluir que a classe 2 é facilmente confundida com a classe 1 e 3. A percentagem de erro do algoritmo de classificação com redes neuronais é neste caso de 12,27%.

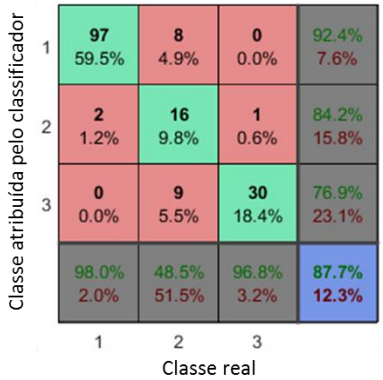


Figura 4.2: Matriz da confusão da rede neuronal gerida pelo Matlab.

Testes com novos jogadores utilizando a Rede Neuronal

Para se testar a rede neuronal foi utilizada a mesma base de dados utilizada para testar as árvores de decisão, onde foram escolhidos os golfistas da Tabela 4.4, e gravadas quarto jogadas para cada tipo de *Swing*, o que dá 80 gravações, 48 *Full Swings*, 16 *Pitchings* de 60 metros e 16 *Pitchings* de 30 metros.

Os resultados do algoritmo de classificação com as redes neuronais com todas as classes são mostrados na Figura 4.3, classificando corretamente 29 amostras em 48 para classe 1 (*Full swing*), as restantes, foram 17 para classe 2 (*Pitching* 60 metros) e 2 para classe 3 (*Pitching* 30 metros). No caso da classe 2, foram acertados 8 amostras em 16, as restantes foram 4 para classe 1 e 4 para classe 3. Por fim, no caso da classe 3, foram acertadas 11 amostras em 16, as restantes foram para classe 2. A classe 1 pode ser facilmente confundida com a classe 2, a classe 2 pode ser confundida tanto com a classe 1 ou a classe 3, por fim a classe 3 é facilmente confundida com a classe 2. A percentagem de erro do classificador para os novos jogadores com algoritmo de classificação com redes neuronais é de 40%, e mais uma vez, o resultado deve-se aos novos golfistas serem de uma faixa etária diferente de aqueles para a qual foi treinada a rede.

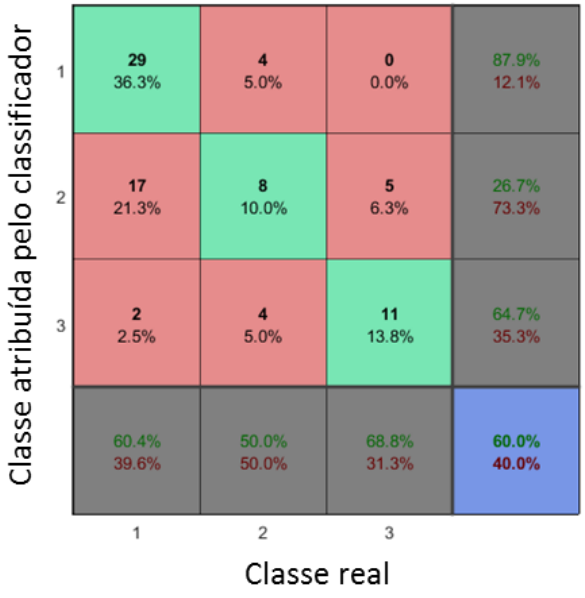


Figura 4.3: Matriz da confusão com os novos jogadores utilizando a rede neural.

4.3 Comparação entre Árvores de Decisão e Redes Neuronais

Nesta aplicação as árvores de decisão conseguem ter melhores resultados em comparação com as redes neuronais (1,84% e 1,23% de erro no caso das árvores de decisão e 12,27% de erro no caso das redes neuronais) para a mesma base de dados.

No entanto, os testes dos algoritmos de classificação com dados novos foram próximos. No caso da árvore de decisão hierárquica com classes separadas foi de 52,5% e 37,5% com todas as classes enquanto que no caso das redes neuronais foi de 40%, (ver Tabela 4.7).

Tabela 4.7 Comparação de resultados.

Classificador	Com os primeiros jogadores	Com os novos jogadores
Árvores de decisão com todas as classes.	1,84%	37,5%
Árvores de decisão com classes separadas.	1,23%	52,5%
Redes Neuronais	12,27%	40%

Podemos portanto concluir que a classificação dos movimentos de golfe com árvores de decisão apresenta melhores resultados quando comparados com as redes neuronais. Por outro lado, a implementação das árvores de decisão é mais fácil e mais prática do que a das redes neuronais.

4.4 Resultado do Algoritmo de Deteção de Jogadas, Extração de características e Classificação

No fim de definir todas as condições para detecção de jogadas, extração de características e criação da árvore de decisão, foi então possível implementar os algoritmos em linguagem C para a detecção automática.

A Figura 4.4 mostra um exemplo de uma resposta de um *Pitching* de 30 metros. O ficheiro com os dados é carregado um a um, isto é, o programa carrega o ficheiro que contém uma só gravação. Para um projeto futuro, estes algoritmos serão implementados no telemóvel inteligente de modo que a detecção, extração de características e a identificação de jogadas de golfe sejam completamente automáticas e em tempo real.

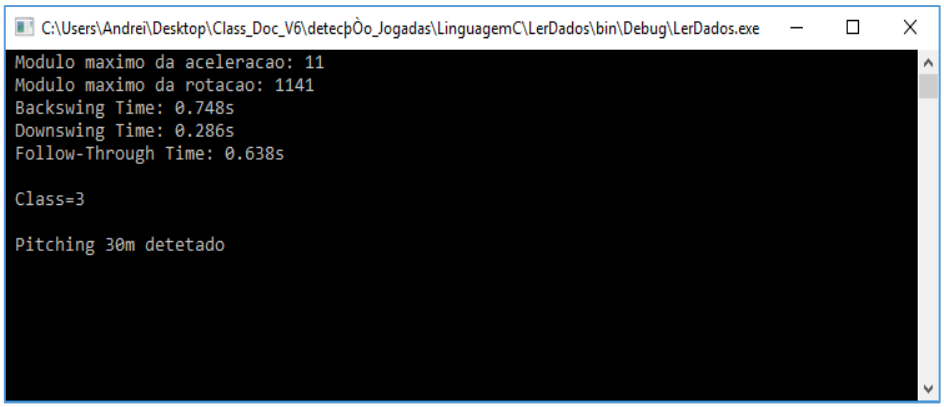


Figura 4.4: Resposta do algoritmo em linguagem C.

5 Conclusões e Trabalhos Futuros

Para concluir, é feita uma análise geral à realização desta dissertação e são ainda apontadas propostas de trabalhos futuros para a melhoria do mesmo.

5.1 Conclusões

Em primeiro lugar, a estrutura da pulseira mostrou ser adequada para os golfistas, tendo um peso e um espaço ocupado no pulso que não prejudica significativamente a realização do movimento do golfe.

Relativamente ao *hardware*, este mostrou-se eficaz, mas verificou-se a necessidade de ter alguns cuidados na utilização da pulseira devido ao material que a envolve ser frágil, levando os fios a dessoldarem-se das placas.

Em seguida, o microprocessador selecionado mostrou bons resultados possibilitando inúmeras funcionalidades e permitindo uma interação prática e de fácil aprendizagem, visto haver uma grande comunidade a desenvolver inúmeras aplicações para o mesmo.

Posteriormente, os algoritmos de classificação mostraram bons resultados, tendo sido o classificador baseado em árvore de decisão considerado a melhor escolha para o problema, devido aos seus resultados e à sua facilidade de implementação.

Este projeto contou com demonstrações em vários locais, tais como o IoT Summit em Lisboa, o aniversário da DNA em Cascais e o aniversário da ESA Bic em Coimbra, onde foram mostradas e discutidas as ideias principais do projeto. Para além disto, a ideia deste projeto foi publicada no portal i9magasine, com o nome da empresa Bluecover.

Para concluir, no geral, as técnicas de aprendizagem automática para a identificação de jogadas de golfe tiveram sucesso, sendo no entanto notória a necessidade de aperfeiçoar os algoritmos de detecção de jogadas e de identificação do tipo do movimento de golfe.

5.2 Trabalhos Futuros

Tendo em consideração as experiências efetuadas com os algoritmos e a análise comportamental à arquitetura da pulseira surge a necessidade de apresentar trabalhos futuros tanto para aperfeiçoamento dos mesmos como para a extensão do projeto.

Os tópicos que se apresentam seguidamente ilustram algumas possíveis melhorias futuras:

- A pulseira poderá ser equipada com um Bluetooth *Low Energy*, de baixo consumo energético, de modo a aumentar a durabilidade da bateria e minimizar a estrutura física tornando o seu uso mais confortável.
- Para o melhoramento da detecção de jogadas de golfe e identificação do tipo de movimento, poderão ser considerados também os dados da aceleração não considerando apenas o valor máximo do módulo de aceleração no momento do batimento do taco de golfe na bola.
- Introduzir funcionalidades para a posição do GPS, como por exemplo, ter a localização do local onde o golfista se encontra durante o jogo.
- Implementar todo o algoritmo no telemóvel inteligente, que constitui a detecção de jogadas, extração de característica e identificação do tipo do movimento de golfe, de modo automático.
- Recolher dados de mais golfistas de diferentes idades, HCPs (*handicap*) e género.
- Desenvolver algoritmos de classificação do tipo de jogada tendo em consideração características adicionais dos jogadores tais como a sua idade e o seu handicap.
- Estudar os movimentos de golfe *Bunker*, *Chippings* e *Putts*.
- Estudar outras aplicações onde se possa utilizar a pulseira.

6 Referências

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2012). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 7657 LNCS, pp. 216–223). https://doi.org/10.1007/978-3-642-35395-6_30

Arduino LIbraries. (2016). LSM6 - Arduino Libraries. Retrieved May 31, 2017, from http://www.arduinolibraries.info/libraries/lsm6

Bunkheila, G. (2016). Signal Processing for Machine Learning. Retrieved May 30, 2017, from https://www.mathworks.com/videos/signal-processing-for-machine-learning-99887.html

Dfrobot. (2016). Bluetooth BEE. Retrieved May 30, 2017, from https://www.dfrobot.com/wiki/index.php/BLUETOOH_BEE_(SKU:TEL0023)

Du, W., Du, W., Zhan, Z., & Zhan, Z. (2002). Building decision tree classifier on private data. *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining-Volume 14*, *14*, 1–8. Retrieved from http://dl.acm.org/citation.cfm?id=850784

Evans, K., & Tuttle, N. (2015). Improving performance in golf: current research and implications from a clinical perspective. *Brazilian Journal of Physical Therapy*, *19*(5), 381–9. https://doi.org/10.1590/bjpt-rbf.2014.0122

Fonseca, J. (1994). *Indução de Árvores de Decisão*. Universidade Nova de Lisboa-FCT.

Golfe Jogar. (2016). A Bola de Golfe. Retrieved May 30, 2017, from http://www.jogargolfe.com/bola-de-golfe.html

Golfeshot. (2016). Golfshot GPS. Retrieved May 30, 2017, from https://golfshotgps.com/pt/

Infopedia. (2016). Golfe. Retrieved May 30, 2017, from https://www.infopedia.pt/$golfe

Nuno Duro. (2015). O meu guia de golfe. Retrieved from https://www.slideshare.net/NunoDuro1/

PTRobotics. (2016a). AltIMU-10 v5. Retrieved May 30, 2017, from https://www.ptrobotics.com/imu/4195-altimu-10-v5-gyro-accelerometer-compass-and-altimeter-lsm6ds33-lis3mdl-and-lps25h-carrier.html

PTRobotics. (2016b). Lithium Polymer USB Charger. Retrieved May 31, 2017, from https://www.ptrobotics.com/lipo/892-lithium-polymer-usb-charger.html?search_query=lithium+charger&results=25

PTRobotics. (2016c). Polymer Lithium Ion Battery. Retrieved May 30, 2017, from https://www.ptrobotics.com/lipo/3231-polymer-lithium-ion-battery-3-7v-850mah.html

PTRobotics. (2016d). Teensy 3.2. Retrieved May 30, 2017, from https://www.ptrobotics.com/freescale/4010-teensy-32.html?search_query=teensy+3.2&results=4

Rules, R. L. (2012). Regras de Golfe, *32ª Edição*. Retrieved from http://portal.fpg.pt/c/document_library/get_file?p_l_id=26329&folderId=20532&name=DLFE-8027.pdf

Salford Systems. (2016). Salford. Retrieved May 30, 2017, from http://info.salford-systems.com/spm-8-download

Santos, T. (2016). *Fuzzy Data Fusion Approach for Remote Sensing Classification.* Universidade Nova de Lisboa-FCT.

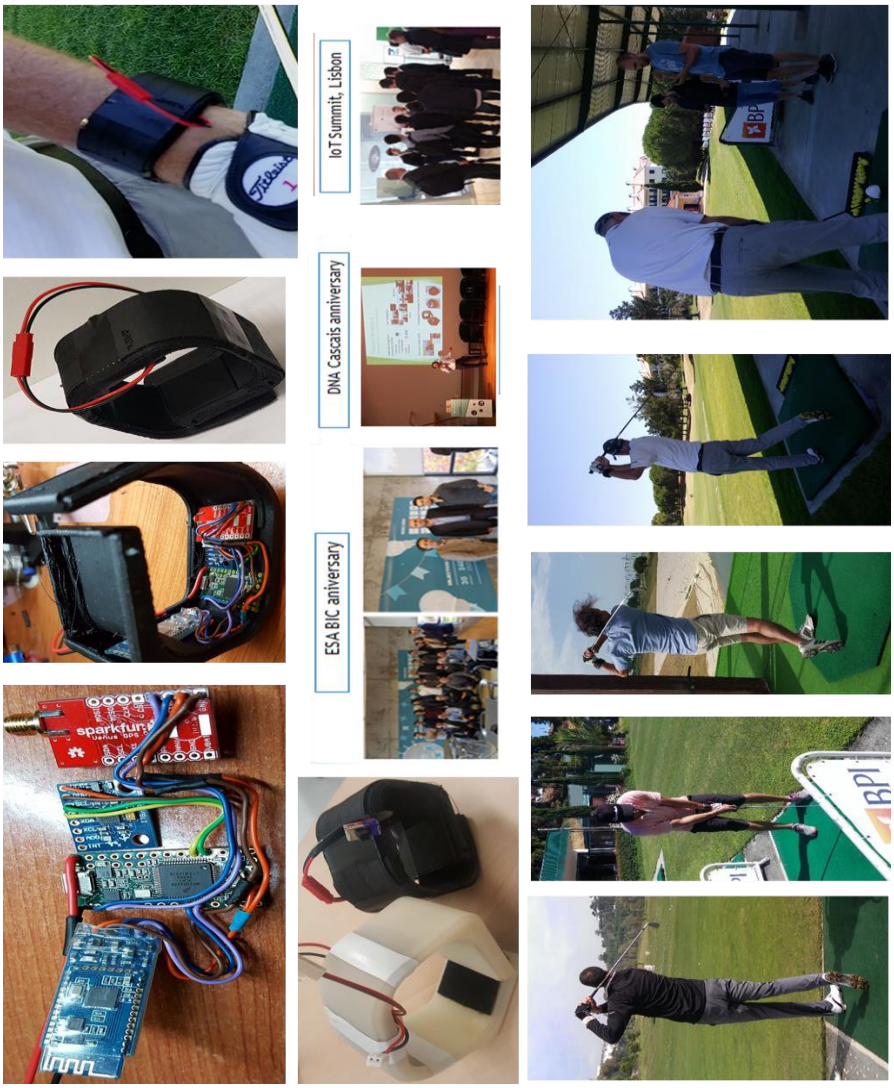
Stančin, S., & Tomažič, S. (2013). Early improper motion detection in golf swings using wearable motion sensors: the first approach. *Sensors (Basel, Switzerland)*, *13*(6), 7505–21. https://doi.org/10.3390/s130607505

Tacadas. (2013). Tacadas de Golfe. Retrieved May 30, 2017, from http://tacadas.com/artigos/saber-basico-sobre-tacos-golfe

TrackMan. (2016). TrackMan Golf. Retrieved May 30, 2017, from https://trackmangolf.com/

Anexos

A. Fotos das Apresentações Públicas



B. Implementação da Pulseira

I. Incluídos

```
#include <Wire.h>
#include <LSM6.h>
#include <LPS.h>
#include <LIS3MDL.h>
#include <Metro.h>

#define BT Serial1 //para gps
#define DUR 20000
#define MAX_RECORD_TIME 999*1000 //maximo tempo permitido
Metro metro(100);
elapsedMillis etime;
#define BT Serial3 // para BLUETOOTH

LSM6 gyro;
LPS ps;
LIS3MDL compass;

char c,report[160];
int sampling_rate=50,Recording;
int run_time=0, LED=13,ts=0;
bool ps_enable, timeout_enable,gyro_enable;
char in,out;
float gyro_values[3], acc_values[3], comp_values[3], ps_values[3];
```

II. Função Setup

```
void setup() {
  BT.begin(38400);
  Serial.begin(9600);
  Wire.begin();
  while (!BT.available()); //fica a espera de algum comando do BT
  while (!BT.read()=='+' && !BT.read()=='+');
  pinMode(LED, OUTPUT);  //definir o a saída para o LED
  imu.writeReg(LSM6::CTRL1_XL, 0x44); //configura o acelerómetro
  imu.writeReg(LSM6::CTRL2_G, 0x4C); //Configura o giroscópio
  gyro_enable=true;
  ps_enable=true;
  Recording=0;
  if (!gyro.init())
      BT.println("Giroscópio e Acelerómetro não encontrado!");
  else
      BT.println("Giroscópio e Acelerómetro encontrado!");
  gyro.enableDefault();
  if (!compass.init())
      BT.println("Magnetómetr não encontrado!");
  else
      BT.println("Magnetómetr encontrado!");
  compass.enableDefault();
  if (!ps.init(ps.device_25H,ps.sa0_high))
      BT.println("Barómetro não encontrado!");
  else
      BT.println("Barómetro encontrado!");
  sampling_rate= 050;
  char d1,d2,d3; //configurar a taxa amostragem
  d1=49;//ascii 49=1
  d2=48;
  d3=48;
  d1=(d1-48)*100;
  d2=(d2-48)*10;
  sampling_rate=d1+d2+(d3-48);
  metro.interval(1000/sampling_rate);
  metro.reset();
  gyro_values[0]=-1.0;
  gyro_values[1]=-1.0;
  gyro_values[2]=-1.0;
  acc_values[0]=-1.0;
  acc_values[1]=-1.0;
  acc_values[2]=-1.0;
  comp_values[3]=-1.0;
  comp_values[4]=-1.0;
  comp_values[5]=-1.0;
  ps_values[0]=-1.0;
  ps_values[1]=-1.0;
  ps_values[2]=-1.0;
   }
```

III. Função Loop

```
void loop() {
  digitalWrite(LED, HIGH); // ligar o led do Teensy
  if (!Recording){
      BT.println("Carregar G ou P para ligar/desligar Giroscópio&Acelerómetro ou Barómetro");
      BT.println("Carregar S para iniciar a gravação e R para parar!!");
      while (getoption());
    }
    if (Recording && !gyro_enable && !accel_enable && !ps_enable) {
      BT.println("Nenhum sensor selecionado");
      while (getoption());
    }
    if (Recording && sampling_rate<0) {
      BT.println("Taza de amostragem nao selecionada");
      while (getoption());
    }
  if (Recording && gyro_enable && accel_enable && metro.check()) {
      gyro.read();// ler os dados
      gyro_values[0]=68.5*gyro.g.x*0.001;
      gyro_values[1]=68.5*gyro.g.y*0.001;
      gyro_values[2]=68.5*gyro.g.z*0.001;
      acc_values[0]=0.000488*gyro.a.x;
      acc_values[1]=0.000488*gyro.a.y+0.04;
      acc_values[2]=0.000488*gyro.a.z;
    }
    if (Recording  && metro.check()) {
      compass.read();
      comp_values[0]=0.16*compass.m.x*0.001;
      comp_values[1]=0.16*compass.m.y*0.001;
      comp_values[2]=0.16*compass.m.z*0.001;
    }
    if (Recording && ps_enable && metro.check()) {
      ps_values[0] = ps.readPressureMillibars();
      ps_values[1] = ps.pressureToAltitudeMeters(ps_values[0]);
      ps_values[2] = ps.readTemperatureC();
    }

    if (Recording && (gyro_enable || accel_enable || ps_enable)) {
      snprintf(report, sizeof(report),"%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.3f",
      gyro_values[0], gyro_values[1], gyro_values[2],
      acc_values[0],acc_values[1],acc_values[2],comp_values[0],comp_values[1],comp_values[2],
      ps_values[0],ps_values[1],ps_values[2],
      etime*0.001);
      BT.println(report);
    }
    if(run_time>0){
      if (Recording && etime>=run_time)
        Recording=1;
    }else{
      if (BT.available()) {
      char d1=BT.read();
      if(d1=='E')
        Recording=0;
      }
      if(etime>=MAX_RECORD_TIME)
        Recording=0;
    }
}
```

IV. Função getoption

```
int getoption(){
  char cc;
  while (!BT.available()); //a espera do comando BT
  cc=BT.read();
  switch (cc) {
    case 'G':
      gyro_enable=!gyro_enable;
      if (gyro_enable) BT.println("Giroscópio e o Acelerómetro ligado");
      else BT.println("Giroscópio e o Acelerómetro desligado");
      break;
    case 'P':
      ps_enable=!ps_enable;
      if (ps_enable) BT.println("Barómetro ligado");
      else BT.println("Barómetro desligado");
      break;
    case 'K':
      char d1,d2,d3;
      if (BT.available()) d1=BT.read();
      else break;
      if (BT.available()) d2=BT.read();
      else break;
      if (BT.available()) d3=BT.read();
      else break;
      if (d1>=48 && d1<=57) d1=(d1-48)*100;
      else break;
      if (d2>=48 && d2<=57) d2=(d2-48)*10;
      else break;
      if (d3>=48 && d3<=57)
      {
        sampling_rate=d1+d2+(d3-48);
        metro.interval(1000/sampling_rate);
        metro.reset();
      }
      else break;
      break;
    case 'R':
      Recording=0;
        BT.println("Para a gravação");
         return 0;
    case 'V':
      Recording=1;
      run_time=0;
      etime=0;
      return 0;
     case 'S':
      Recording=1;
       BT.println("Inicia a gravação");
      char v1,v2,v3;
      if (BT.available()) v1=BT.read();
      else break;
```

```
        if (BT.available()) v1=BT.read();
        else break;
        if (BT.available()) v2=BT.read();
        else break;
         if (BT.available()) v3=BT.read();
        else break;
        if (v1>=48 && v1<=57) v1=(v1-48)*100;
        else break;
        if (v2>=48 && v2<=57) v2=(v2-48)*10;
        else break;
        if (v3>=48 && v3<=57) v3=(v3-48);
        else break;
        run_time=v3+v2+v1;
        run_time=run_time*1000;
        etime=0;
        return 0;
      }
   return 1;
}
```

C. Algoritmo de Deteção e Extração

```
#include <stdio.h>
#include <stdlib.h>

int classSwing(int maxA, int maxG,float backtime,float downtime,float followtime);
void printClass(int classSwing);
int main()
{
    int modAcc[700],GyroX[700],GyroY[700],GyroZ[700], modGyro[700];
    int temp,temp1, maxG=0,maxA=0, maxp,ib, val=0;
    float downtime=0, backtime=0, followtime=0;
    int i=0, tam=0,j=0;
    FILE * fp;
    fp = fopen ("file.txt", "r");
    // carrega o vetor e calcula o tamanho do mesmo
    while(fscanf(fp," %d %d %d %d %d",&GyroX[i],&GyroY[i],&GyroZ[i],&modAcc[i],&modGyro[i])!=EOF){
       tam+=1;
       i++;
       }
    //procura o maximo do modulo da rotação e o ponto do mesmo
    for(i=0;i<tam;i++){
        if(maxG<modGyro[i]){
            maxG=modGyro[i];
            maxp=i+1;
        }
    }
    //procura o maximo do modulo da Aceleração
    for(i=maxp-2;i<maxp+2;i++){
        if(maxA<modAcc[i]){
            maxA=modAcc[i];
        }
    }
```

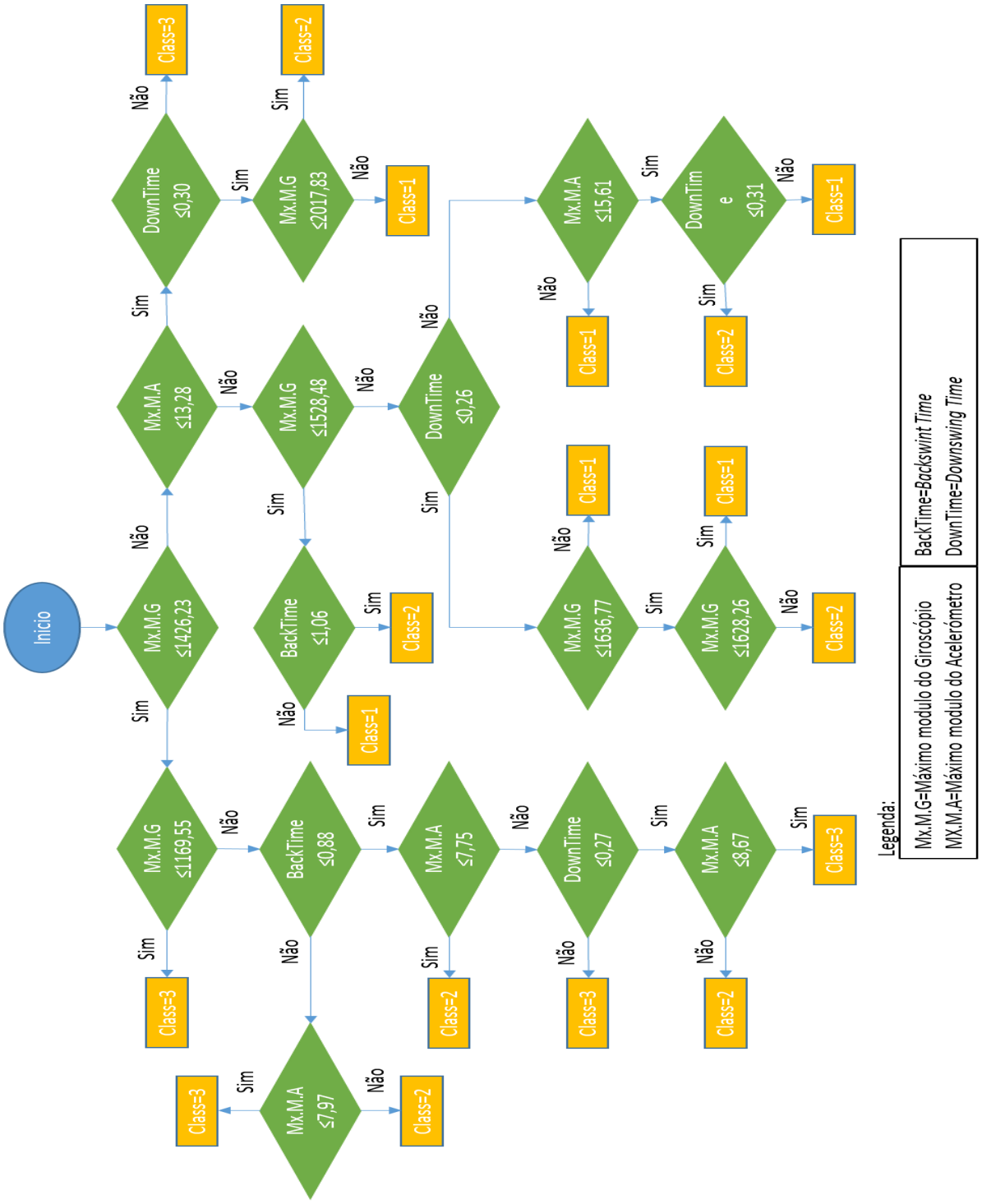
```
//detectar full swing e pitching 60m
    if(maxG>1150&&maxA>7&&tam>160){
        //Verifica o valor do eixo X do giro., que está no mesmo ponto que o mód. máx. da rotação
        temp=0.99*maxG, temp1=0.8*maxG;
        if(GyroX[maxp-1]<temp&&GyroX[maxp-1]>temp1)
            val=1;
        else
            val=0;
    if(val==1){
        //Verifica o valor do eixo Z do giro., que está depois dos mód. máx. da rotação.
        temp=-0.1*maxG, temp1=-0.4*maxG;
        for(i=maxp+1;i<=maxp+6;i++){
            if(GyroZ[i]<temp&&GyroZ[i]>temp1){
                val=1;
                break;
            }
            val=0;
        }}
    if(val==1){
        //Verifica o valor do eixo Y do giro., que está depois dos máximos dos módulos.
        temp=0.05*maxG, temp1=0.4*maxG;
        for(i=maxp+1;i<=maxp+6;i++){
            if(GyroY[i]>temp&&GyroY[i]<temp1){
                val=1;
                break;
            }
            val=0;
        }}
    //Verifica o valor do eixo Z do giro., que está antes dos mód. máx. da rotação
    temp=0.5*maxG, temp1=0.30*maxG;
    if(val==1)
        for(i=maxp-3;i>=maxp-6;i--){
            if(GyroZ[i]<temp&&GyroZ[i]>temp1){
                val=1;
                break;
            }
            val=0;}
    if(val==1){
        for(i=maxp-10;i>=maxp-20;i--){   //calcula o downswing time
            if(modGyro[i]<70){
                downtime=(maxp-i)*0.022;
                //Verifica a inversão do sinal X, Y e Z
                if(GyroZ[i-3]<0&&GyroZ[i+3]>0&&GyroX[i-5]<0&&GyroX[i+5]>0&&GyroY[i-3]>0&&GyroY[i+3]<0)
                    val=1;
                else
                    val=0;
                break;}
            val=0;
        } }
    if(val==1){//calcula o backswng time
        for(ib=i-37;ib>=i-67&&ib>=0;ib--){
            if(modGyro[ib]<16){
                backtime=(i-ib-1)*0.022;
                //Verifica a inversão do sinal X e Z
                if(GyroZ[ib+15]<0&&GyroX[ib+10]<0)
                    val=1;
                else
                    val=0;
                break;}
            val=0;}}
```

```
            val=0;}}
    //calcula o backswing time quando a pessoa meixa muito o taco
    if(val==0&&downtime!=0){
        for(ib=i-37;i>=i-67;ib--){
            if(modGyro[ib]<25){
                backtime=(i-ib-3)*0.022;
                //Verifica a inversão do sinal X e Z
                if(GyroZ[ib+15]<0&&GyroX[ib+10]<0)
                    val=1;
                else
                    val=0;
                break;}
            val=0;}
    }
    if(val==1){//calcula o Follow-Through  Time
        for(ib=maxp+23;ib<maxp+74&&ib<tam;ib++){
            if(modGyro[ib]<30){
                followtime=(ib-maxp+1)*0.022;
                val=1;
                break;}
            val=0;}
            if(val==0&&backtime!=0){
            for(ib=maxp+23;ib<maxp+74&&ib<tam;ib++){
            if(modGyro[ib]<71){
                followtime=(ib-maxp)*0.022;
                val=1;
                break;}
            val=0;}
        }}}

    //detectar pitching 30m
    if(maxG<=1150&&maxG>840&&maxA>5&&maxA<=19&&tam>120){
        //Verifica o valor do eixo X do giro, que está no mesmo ponto que o mód. máx. da rotação
        temp=0.99*maxG, temp1=0.8*maxG;
        if(GyroX[maxp-1]<temp&&GyroX[maxp-1]>temp1)
            val=1;
        else
            val=0;
        if(val==1){
            //Verifica o valor do eixo Z do giro., que está depois dos mód. máx. da rotação.
            temp=-0.1*maxG, temp1=-0.4*maxG;
            for(i=maxp+1;i<=maxp+6;i++){
                if(GyroZ[i]<temp&&GyroZ[i]>temp1){
                    val=1;
                    break;}
                val=0;}}
        if(val==1){
            //Verifica o valor do eixo Y do giro., que está depois dos máximos dos módulos.
            temp=0.05*maxG, temp1=0.4*maxG;
            for(i=maxp+1;i<=maxp+6;i++){
                if(GyroY[i]>temp&&GyroY[i]<temp1){
                    val=1;
                    break;}
                val=0;}}
         //Verifica o valor do eixo Z do giro., que está antes dos máximos do módulo da rotação
        temp=0.5*maxG, temp1=0.30*maxG;
        if(val==1)
            for(i=maxp-3;i>=maxp-6;i--){
                if(GyroZ[i]<temp&&GyroZ[i]>temp1){
                    val=1;
                    break;}
```

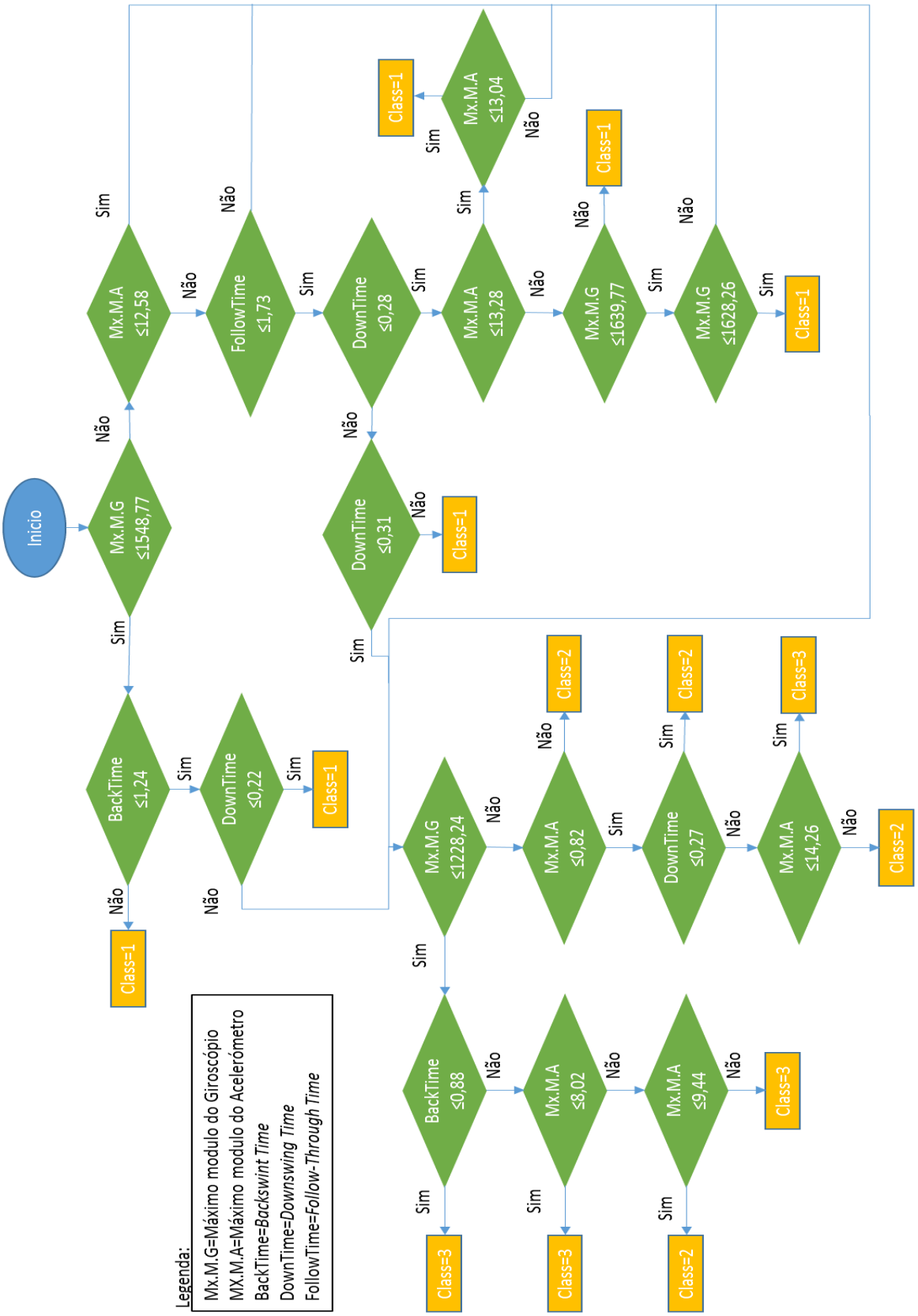
```
                    break;}
                val=0;}
            if(val==1){
                for(i=maxp-10;i>=maxp-23;i--){   //calcula o downswing time
                    if(modGyro[i]<45){
                        downtime=(maxp-i-2)*0.022;
                        //Verifica a inversão do sinal X, Y e Z
                        if(GyroZ[i-3]<0&&GyroZ[i+3]>0&&GyroX[i-5]<0&&GyroX[i+5]>0&&GyroY[i-3]>0&&GyroY[i+3]<0)
                            val=1;
                        else
                            val=0;
                        break;}
                    val=0;}}
            if(val==1){//calcula o backswing time
                for(ib=i-29;ib>=i-45&&ib>=0;ib--){
                    if(modGyro[ib]<16){
                        backtime=(i-ib-2)*0.022;
                        //Verifica a inversão do sinal X e Z
                        if(GyroZ[ib+15]<0&&GyroX[ib+10]<0)
                            val=1;
                        else
                            val=0;
                         break;}
                    val=0;}}
            if(val==0&&downtime!=0){ //calcula o backswing time quando a pessoa meixa muito o taco
                for(ib=i-29;i>=i-45;ib--){
                    if(modGyro[ib]<26){
                        backtime=(i-ib-2)*0.022;
                        //Verifica a inversão do sinal X e Z
                        if(GyroZ[ib+15]<0&&GyroX[ib+10]<0)
                          val=1;
                           backtime=(i-ib-2)*0.022;
                           //Verifica a inversão do sinal X e Z
                           if(GyroZ[ib+15]<0&&GyroX[ib+10]<0)
                             val=1;
                           else
                             val=0;
                           break;}
                    val=0;}}
             if(val==1){//calcula o Follow-Through  Time
                 for(ib=maxp+20;ib<maxp+55&&ib<tam;ib++){
                     if(modGyro[ib]<30){
                       followtime=(ib-maxp)*0.022;
                       val=1;
                       break;}
                     val=0;}
                 if(val==0&&backtime!=0){
                     for(ib=maxp+20;ib<maxp+55&&ib<tam;ib++){
                         if(modGyro[ib]<60){
                           followtime=(ib-maxp)*0.022;
                           val=1;
                           break;}
                         val=0;}}}
}


    if(val==1){
        printf(" Modulo maximo da aceleracao: %d\n",maxA);
        printf(" Modulo maximo da rotacao: %d\n",maxG);
        printf(" Backswing Time: %.3fs\n",backtime);
        printf(" Downswing Time: %.3fs\n",downtime);
        printf(" Follow-Through Time: %.3fs\n",followtime);
        printClass(classSwing(maxA,maxG,backtime,downtime,followtime));
        }
    else
        printf(" \nNao foi detectada a jogada!\n\n\n\n\n");
        fclose(fp);
}
```

D. Árvore de Decisão

I. Fluxograma com todas as classes



II. Fluxograma hierárquica com classes separadas



III. Árvore de decisão no Excel

1- Com todas as classes

B2=Max.Mod.Aceleração; C2=Max.Mod.Rotação; D2=*Backswing Time*; E2=*Downswing time*
=SE(C2<=1426,23; SE(C2<=1169,55; 3; SE(D2<=0,88; SE(B2<=7,75; 2; SE(E2<=0,27; SE(B2<=8,67;3;2);3));SE(B2<=7,97;3;2)));SE(B2<=13,28;SE(E2<=0,3;SE(C2<=2017,83;2;1); 3);SE(C2<=1528,48;SE(D2<=1,06;2;1);SE(E2<=0,26;SE(C2<=1636,77;SE(C2<=1628,26;1;2) ;1); SE(B2<=15,61;SE(E2<=0,31;2;1);1)))))

2- Com classes separadas

B2=Max.Mod.Aceleração; C2=Max.Mod.Rotação; D2=*Backswing Time*; E2=*Downswing time;* F2=*Follow-Through Time*
=SE((SE(C2<=1548,77;SE(D2<=1,24;SE(E2<=0,22;1;2);1);SE(B2<=12,58;2;SE(F2<=1,73; SE(E2<=0,28;SE(B2<=13,28;SE(B2<=13,04;1;2);SE(C2<=1639,77;SE(C2<=1628,26;1;2);1));SE(E2<=0,31;2;1));2))))=2;(SE(C2<=1228,24;SE(D2<=0,88;3;SE(B2<=8,02;3;SE(B2<=9,44 ;2;3))); SE(F2<=0,82; SE(E2<=0,27;2; SE(B2<=14,26;3;2));2)));1)

IV. Árvore de decisão em Linguagem C

```
int classSwing(int maxA, int maxG,float backtime,float downtime,float followtime){
    int classe=0;
    if(maxG<=1548.77)
        if(backtime<=1.24)
            if(downtime<=0.22)
                classe=1;
            else
                classe=2;
        else
            classe=1;
    else
        if(maxA<=12.58)
            classe=2;
        else
            if(followtime<=1.73)
                if(downtime<=0.28)
                    if(maxA<=13.28)
                        if(maxA<=13.04)
                            classe=1;
                        else
                            classe=2;
                    else
                        if(maxG<=1639.77)
                            if(maxG<=1628.26)
                                classe=1;
                              else
                                  classe=2;
                             else
                                 classe=1;
                else
                    if(downtime<=0.31)
                        classe=2;
                    else
                        classe=1;
            else
                classe=2;

     if(classe==2){
        if(maxG<=1228.24)
            if(backtime<=0.88)
                classe=3;
            else
                if(maxA<=8.02)
                    classe=3;
                else
                    if(maxA<=9.44)
                        classe=2;
                    else
                        classe=3;
        else
```

```
                if(followtime<=0.82)
                    if(downtime<=0.27)
                        classe=2;
                    else
                        if(maxA<=14.26)
                            classe=3;
                        else
                            classe=2;
                else
                    classe=2;

        }


    printf("\n Class=%d\n",classe);
        return classe;
    }

    void printClass(int classSwing){
     switch(classSwing){
                case 1 : printf("\n Full-Swing detetado\n"); break;
                case 2 : printf("\n Pitching 60m detetado\n"); break;
                case 3 : printf("\n Pitching 30m detetado\n"); break;
              default : printf(" Nenhuma classe encontrada\n");
        }}
```

E. Rede Neuronal

```
%   Treino - input data.
%   Classes - target data.
x = Treino;
t = Classes;
trainFcn = 'trainscg';  % Scaled conjugate gradient backpropagation.
% Create a Pattern Recognition Network
hiddenLayerSize = 10;
net = patternnet(hiddenLayerSize);
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 90/100;
net.divideParam.valRatio = 5/100;
net.divideParam.testRatio = 5/100;
net.performFcn = 'crossentropy';  % Cross-Entropy
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
    'plotconfusion', 'plotroc'};
% Train the Network
[net,tr] = train(net,x,t);
% Test the Network
```

```
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)
tind = vec2ind(t);
yind = vec2ind(y);
percentErrors = sum(tind ~= yind)/numel(tind);
% Recalculate Training, Validation and Test Performance
trainTargets = t .* tr.trainMask{1};
valTargets = t .* tr.valMask{1};
testTargets = t .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)
% View the Network
view(net)
% Change the (false) values to (true) to enable the following code blocks.
% See the help for each generation function for more information.
if (false)
    genFunction(net,'myNeuralNetworkFunction');
    y = myNeuralNetworkFunction(x);
end
if (false)
    genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
    y = myNeuralNetworkFunction(x);
end
if (false)
    gensim(net);
end
function [y1] = myNeuralNetworkFunction(x1)
% ===== NEURAL NETWORK CONSTANTS =====
% Input 1
x1_step1_xoffset = [5.246;847.509;0.641;0.194;0.453];
x1_step1_gain =
[0.0971911750413062;0.00150031656679559;2.29095074455899;0.968992248062015;1
.54559505409583];
x1_step1_ymin = -1;

% Layer 1
b1 = [1.2375479721152978;2.2321501009999754;1.1508168915503647; ...
      -0.89235368999320908;0.37019248515664854;0.094854568894959676; ...
      -0.93905419522346967;1.2387669407478372;-1.8006898931128819;-
1.8901468778052877];
IW1_1 = [0.4663367212465378 -2.5223682919893569 -0.16380991822575761 ...
    2.2671859404779804 -0.66757495608089712;-0.50772387875220637 -
0.70956182877922513 ...
    -0.38486901384861116 0.75300020289147107 -1.4320532064908806;-
0.77586285100051855 ...
    -0.358807582218581 -1.2364500479207137 0.27374051986326775
1.9851107011037934; ...
    1.0930193382025966 -0.90895573364605431 0.72446614660042807
1.2813313521179155 ...
    -0.92364576934221454;0.2839824031367178 -0.079093053148768452 -
1.145648146001216 ...
    -2.0251081124654555 -0.010983766114003545;-2.4638355152718772 -
1.4414572631542086 ...
    -0.47605145391378567 0.016355740607359295 -0.070475282801994152;-
0.87505321448994133 ...
    -1.0258442043913829 0.93557464607409024 0.92787512440277387
1.3578815160884901; ...
```

```
    1.6901356541706853 0.28003707681179585 1.0016538960809198
0.94495086418592633 ...
    -0.24652897123830328;0.47483581168108491 -0.025175875896367196 -
1.7980337420565757 ...
    -0.34164299688597177 -1.6189955384232746;-1.1639028819082229 -
1.6922473922100141 ...
    0.5355082927398287 -0.12484062616835183 1.5023094011105327];

% Layer 2
b2 = [0.37538573085990989;-0.082383630142039657;-0.87185175687303074];
LW2_1 = [-2.5056619600290779 0.21227539801878681 0.30506650733055962
0.88779515024659716 ...
    -0.077595632207392892 -3.582139449608452 0.43397006476624855
1.7609221783854363 ...
    -1.364092210654376 -0.35211455274303188;-1.7298666117977892 -
0.35975826385510129 ...
    -0.48332560754788362 -0.0080849287069941801 -0.20273748085400467
0.30723192034608665 ...
    -0.012309542231507441 0.69242901434017179 -1.4665125471496996 -
0.16634999730864822; ...
    2.0400833064721002 -0.53173352599750379 1.1493375515495174
0.32311291126203373 ...
    -0.54102365443950806 1.3238661678111894 0.91413383870664811 -
0.70091600887047523 ...
    1.0071560675942464 1.0660108014104253];

% ===== SIMULATION ========
% Dimensions
Q = size(x1,2);
% Input 1
xp1 = mapminmax_apply(x1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);
% Layer 1
a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);
% Layer 2
a2 = softmax_apply(repmat(b2,1,Q) + LW2_1*a1);
% Output 1
y1 = a2;
end
% ===== MODULE FUNCTIONS ========
% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
y = bsxfun(@minus,x,settings_xoffset);
y = bsxfun(@times,y,settings_gain);
y = bsxfun(@plus,y,settings_ymin);
end
% Competitive Soft Transfer Function
function a = softmax_apply(n)
nmax = max(n,[],1);
n = bsxfun(@minus,n,nmax);
numer = exp(n);
denom = sum(numer,1);
denom(denom == 0) = 1;
a = bsxfun(@rdivide,numer,denom);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
a = 2 ./ (1 + exp(-2*n)) - 1;
end
```